НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_004.415.25____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

Магістерська дисертація

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем
на тему: «Android додаток управління педагогічними та науковими аспектами роботи кафедри»

Виконав: студент _6__курсу, групи_ТВ-71мп__
___Гуменний Аркадій Андрійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _к.т.н, доцент. Карпенко Є.Ю._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент___ _____
(посада, вчене звання, **науковий** ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2018

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий рівень

Спеціальність “Програмне забезпечення розподілених систем”

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)
«____» _____ 2018р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Гуменному Аркадію Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Android додаток управління педагогічними та науковими аспектами роботи кафедри
Науковий керівник Карпенко Євген Юрійович, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 15 березня 2018 року №1151-с
2. Строк подання студентом дисертації _____
3. Об’єкт дослідження програмне забезпечення автоматизованих систем
4. Предмет дослідження програмне забезпечення управління педагогічними та науковими аспектами кафедри _____
5. Перелік питань, які потрібно розробити аналіз існуючих засобів для управління роботою кафедри; дослідження нативних методів розробки під android; дослідження кросплатформних методів розробки під android; дослідження архітектури android додатків; удосконалення архітектури android додатків; розробка програмного продукту для управління педагогічними та науковими аспектами кафедри.
6. Перелік ілюстративного матеріалу схеми архітектури, діаграми функціоналу, інтерфейс.

7. Перелік публікацій тези доповіді XVI міжнародної науково-практичної конференції аспірантів, магістрів і студентів (Київ, 24-27 квітня 2018 р); тези доповіді II всеукраїнської студентської науково-практичної конференції «Наука XXI століття: виклики, пріоритети, перспективи досліджень» (22 березня 2018 р.); стаття в колективній монографії для V Міжнародної науково-практичної конференції «Сталий розвиток – XXI століття: управління, технології, моделі (наукові читання імені Ігоря Недіна)»

8. Дата видачі завдання « 11 » вересня 2017 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз проблеми управління педагогічними аспектами кафедри	11.09.2017-12.11.2017	
2	Аналіз існуючих реалізацій та шляхів вирішення проблеми	13.11.2017-15.01.2018	
3	Аналіз нативних методів розробки	16.01.2018-05.03.2018	
4	Аналіз кросплатформних методів розробки	06.03.2018-27.06.2018	
5	Аналіз вимог до системи	28.06.2018-18.08.2018	
6	Моделювання схеми роботи майбутньої програми	19.08.2018-30.08.2018	
7	Розробка архітектури програмного забезпечення	01.09.2018-20.09.2018	
8	Розробка дизайну API та серверної частини	21.09.2018-10.10.2018	
9	Розробка мобільного застосунку	11.10.2018-30.11.2018	
10	Оформлення документації	01.12.2018-07.12.2018	

Студент _____ _____
(підпис) (прізвище та ініціали)

Науковий керівник _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Структура й обсяг дипломної роботи.

Магістерська дисертація складається зі вступу, п'яти розділів, висновку, переліку посилань з 67 найменувань, 2 додатки, і містить 20 рисунки, 22 таблиці. Повний обсяг магістерської дисертації складає 96 сторінок, з яких перелік посилань займає 5 сторінок, додатки — 19 сторінок.

Актуальність теми. У дипломі піднято дві теми: автоматизація одного з робочих процесів на кафедрі та дослідження засобів розробки із ціллю їх удосконалення для полегшення написання програмних засобів. Обидві теми є актуальними, так як:

— державній сфері бракує багатьох інструментів, що щорічно полегшували б життя людям;

— сфера розробки постійно розвивається, програмісти постійно досліджують набутки світу розробки задля полегшення своєї роботи та покращення її якості.

Мета дослідження полягає у визначенні способів пришвидишення розробки та підтримки мобільних додатків а також підвищення їх якості за рахунок удосконалення існуючих методів та засобів розробки.

Для досягнення поставленої задачі були сформульовані наступні завдання дослідження, що визначили логіку дослідження та його структуру:

— проаналізувати існуючі засоби розробки під Android (нативні та кросплатформні);

— проаналізувати існуючі універсальні методи, патерни та принципи розробки програмного забезпечення;

— удосконалити підхід до розробки додатків під Android з урахуванням особливостей платформи;

— розробити програмний продукт для автоматизації одного з бізнес-процесів на кафедрі з урахуванням досліджених та удосконалених методів розробки.

Об’єктом дослідження є методи, патерни та принципи розробки а також платформа android.

Предметом дослідження є нові засоби розробки під android, що спиратимуться та будуть удосконалювати існуючі засоби.

Методи дослідження. Розв’язання поставлених задач виконувались засобами комп’ютерного програмування та вивчення напрацювань світової спільноти програмістів, зокрема були вивчені методи розробки:

— нативне програмування із використанням патернів MVP, MVC, MVVM;

— кросплатформне програмування.

Наукова новизна одержаних результатів. Найбільш суттєвими науковими результатами магістерської дисертації є:

— удосконалено спосіб проектування архітектури додатків під платформу android, що дозволяє збільшити швидкість розробки мобільних застосунків;

— набуло подальшого розвитку використання патернів та існуючих засобів написання програмного коду для вирішення питань стабільного нарощення функціоналу із легкою підтримкою, а також можливістю стабільного випуску оновлень додатків.

Практичне значення одержаних результатів роботи полягає в розробці програмних засобів, які полегшать розробку мобільних додатків на ОС Android а також розробка додатку що автоматизує один з робочих процесів на кафедрі.

Ключові слова. *ПАТЕРН, АРХІТЕКТУРА, МОБІЛЬНИЙ ДОДАТОК, ANDROID, НАТИВНИЙ, КРОСПЛАТФОРМНИЙ.*

ABSTRACT

The structure and volume of the thesis.

Master's thesis consists of an introduction, five chapters, conclusion, list of references with 67 titles, 2 annexes, and contains 20 figures, 22 tables. The full range of master's thesis is 96 pages with a list of links takes 5 pages, apps — 19 pages.

Topicality of the theme. Two topics were raised in the diploma: the automation of one of the work processes at the department and the study of development tools with the aim of improving them to facilitate the writing of software. Both themes are relevant, since:

— the state sector lacks many instruments that would make people's lives easier each year;

— the field of development is constantly evolving, programmers are constantly exploring the achievements of the world of development in order to facilitate their work and improve its quality.

The purpose and problems of research is to identify ways to enhance the development and support of mobile applications and improve their quality by improving existing methods and tools.

To accomplish the task, the following research objectives were formulated, which determined the logic of the research and its structure:

— analyze existing Android development tools (native and cross-platform);

— analyze existing universal methods, patterns and principles of software development;

— improve the approach to developing applications for Android, taking into account the features of the platform;

— develop a software product for automating one of the business processes at the department, taking into account the research and advanced methods of development.

The object of the research is the methods, patterns and principles of development, as well as the platform android.

The subject of the research is the new android development tools that will build on and improve the existing tools.

The solution of the set tasks was performed by means of computer programming and studying of the developments of the global community of programmers, in particular, the methods of development were studied:

— native programming using MVP, MVC, MVVM patterns;

— cross-platform programming.

Scientific novelty of the results. The most significant scientific results of the master's thesis are:

— improved the way to design an application architecture for the android platform, which allows you to increase the speed of developing mobile applications;

— got further development in use of patterns and existing tools for writing code to resolve issues of stable build-up of functional with easy support, as well as the possibility of stable release of application updates.

The practical value of the results of the work is to develop software that will facilitate the development of mobile applications on the Android OS, as well as the development of an application that automates one of the work processes at the department.

Keywords. PATTERN, ARCHITECTURE, MOBILE APPLICATION, ANDROID, NATIVE, CROSSPLATFORM.

ЗМІСТ

ВСТУП

Автоматизація робочих процесів — один з основних напрямів у процесі сталого розвитку науки, освіти, та промисловості. Все більше підприємств та державних організацій переходять на електронну документацію а також створюють веб-додатки для забезпечення розподіленого управління бізнес процесами. Останнім часом із розвитком комп'ютерних технологій та інтернету поширюється використання саме мобільних смартфонів для забезпечення діяльності, описаної вище, так як телефони мають значні переваги над ПК, серед яких не тільки мобільність, але й поширеність та доступність.

Як результат, на сьогодні мобільні додатки покривають майже всі сфери ділового та світського життя людини — від інтернет-банкінгу до месенджерів.

Але сфера освіти ще не повністю автоматизована, а деякі процеси в вузі є або занадто бюрократичними, або недостатньо уніфіковані в плані організації їх проведення. Серед таких процесів є пошук студентами викладачів для ведення дипломної роботи. З даним процесом зіштовхуються усі студенти. Під час цього процесу студенти зіштовхуються з наступними проблемами:

— пошук викладача із яким було б приємно вести роботу;

— визначення завантаженості викладача;

— визначення напрямів, за якими викладач має можливість бути дипломним керівником;

— узгодження теми з викладачем.

Для вирішення всіх цих питань необхідний безпосередній особистий контакт із викладачем. Пошук викладача на робочому місці займає час, який можна було б витратити ефективно на виконання дипломної роботи. Враховуючи те, що кожний з вище перерахованих кроків може закінчитися невдало, студенту доведеться все починати спочатку і витрачати ще більше часу. В той самий час, викладач зазвичай є автором ідей своїх дипломних робіт, і публічне оголошення ідей, а також даних про свою завантаженість, могли б полегшити процес пошуку викладачів.

Тому актуальною задачею є розробка сервісу для автоматизації процесу пошуку дипломних керівників та узгодження з ними теми дипломної роботи.

Метою даної роботи є реалізація мобільного додатку для управління педагогічними та науковими аспектами кафедри.

Завдання полягає у створенні мобільного додатку для смартфонів на базі ОС Android, який дозволяє користувачам налагодити процес взаємодії один з одним у контексті вирішення питань щодо організації наукової діяльності кафедри.

Для комунікації між окремими android-пристроями, необхідно також створити сервер, що буде зберігати інформацію користувачів. Таким чином, мобільні пристрої будуть виступати в ролі клієнтів, а запропоноване рішення буде мати клієнт-серверну архітектуру[1].

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК	- персональний комп’ютер
ПП	- програмний продукт
UI	- інтерфейс користувача
ООП	- об’єктно-орієнтоване програмування
ОС	- операційна система
НТУУ “КПІ”	- Національний технічний університет України "Київський політехнічний інститут"
UX	- досвід користувача

1 ЗАДАЧА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ УПРАВЛІННЯ ПЕДАГОГІЧНИМИ ТА НАУКОВИМИ АСПЕКТАМИ КАФЕДРИ

Метою є розробка мобільного додатку для управління педагогічними та науковими аспектами кафедри.

Для досягнення поставленої мети необхідно:

— провести аналіз предметної області;

— виділити сутності предметної області;

— виділити функціонал системи;

— реалізувати обрані функції;

— проаналізувати вже існуючі модулі та програмні продукти, що відповідають функціям майбутньої системи;

— обрати засоби розробки системи;

— реалізувати всі функції системи, які поставленні замовником.

Програмне забезпечення повинно мати наступні функції:

— авторизація;

— управління профілем (перегляд, редагування);

— перегляд інформації про наукові лабораторії;

— перегляд інформації про викладачів;

— управління дипломними роботами (створення, запропонування, відстеження);

— перегляд графіків виконання робіт;

— захист приватних даних користувача системи.

Даний сервіс складається з двох модулів — сервер для зберігання та обробки даних користувачів і мобільний додаток, що дозволяє оперувати даними, що знаходяться на сервері (створювати, редагувати та видаляти дані, проводити пошук та відтворення через користувацький інтерфейс).

Вхідними даними є дії користувача, що направлені на взаємодію із сутностями маршрутів та користувачів (створення, пошук, перегляд). Дані зберігаються на віддаленому сервері у реляційній базі даних, збереження даних та встановлення зв’язків між сутностями знаходиться на віддаленому сервері, а задання параметрів — у клієнтському додатку.

Вихідна інформація — сутності, що було створено на віддаленому сервері, запит і відображення яких відбуваються у клієнтському додатку.

Потенційні користувачі програмного забезпечення, що розробляється — це студенти та викладачі кафедри АПЕПС (а також можливо й інших кафедр).

Висновки до розділу 1

У розділі було висвітлено ідею магістерської дисертації, висвітлено проблеми існуючого програмного забезпечення у контексті предметної області. Також були визначені кроки для реалізації ідеї та оголошені вимоги до розроблюваної програми, коротко описані модулі програми, визначені поняття вхідних та вихідних даних а також потенційні користувачі програмного комплексу.

2 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Аналіз існуючих рішень є необхідним для визначення сильних та слабких сторін розроблюваного програмного комплексу.

2.1 Аналіз існуючих програмних засобів

Ідея організації наукової роботи не нова, і у світі представлена велика кількість подібних сервісів. Кожен рік з’являються десятки аналогічних рішень. Основною проблемою існуючих сервісів є орієнтація на веб платформу та фокусування на конкретному вузі чи кафедрі. Мобільних додатків для організації наукової діяльності ще не було.

Одним з подібних сервісів є Campus KPI[2] – веб-сервіс для студентів та викладачів КПІ. Даний сервіс має широкий функціонал, що частково перетинається з функціоналом даного програмного продукту.

В кампусі у кожного студента та викладача є свій обліковий запис та профіль, за допомогою якого користувачі взаємодіють між собою. Зазвичай кампус використовується для афішування новин від викладачів та для поширення учбових матеріалів (методичок, книжок, завдань лабораторних робіт та інше).

Даний сервіс, у свою чергу, направлений на інший аспект студентського життя – на наукову роботу, а саме на організацію процесу ведення дипломної роботи від визначення керівника до узгодження графіку виконання роботи.

До інших особливостей даного продукту відноситься те, що він орієнтований саме на мобільну платформу, що дозволяє мати актуальну інформацію у будь-який час доби.

І викладачі, і студенти є власниками смартфонів. Студенти надають перевагу електронному спілкування саме через телефони. Також вони поширюють матеріали між собою у месенджерах і слідкують за новинами у соц.мережах. Але коли

студенти намагаються зв’язатися з викладачами, то мають два шляхи – або зустріти викладача на кафедрі, або писати у соц-мережах чи на електронну пошту. Електронна пошта не підходить для оперативного листування, так як користувачі зазвичай перевіряють її один раз на добу, або рідше. Вести ділове спілкування у соц. мережах викладачам також не хочеться, бо це перетин формальних та неформальних відносин між студентом, і тому з'являються такі сервіси як Кампус чи даний програмний продукт.

Але частота відвідування кампусу ще нижча, ніж електронної пошти, тому кампус не підходить для ведення спілкування. На відміну від кампуса, даний продукт орієнтований на мобільні телефони, таким чином і студент, і викладач завжди мають актуальну інформацію щодо статусів дипломних робіт і можуть спілкуватися, не хвилюючись що їх не почують.

2.2 Аналіз існуючих методів розробки програмних засобів

В рамках дисертації були виконані дослідження способів розробки мобільних додатків для Android[3]. Було проаналізовано декілька способів розробки, в тому числі:

— використання засобів нативної розробки;

— використання кросплатформних рішень із альтернативною структурою фреймворку.

Під нативними розуміються засоби, за допомогою створюються програми, які відповідають вимогам певної операційної системи за допомогою її SDK (а також апаратної пам'яті, передач та інших програм, встановлених на пристрої)[4].

Було визначено, що до переваг нативних додатків відноситься:

— швидкодія додатку;

— повноцінний UX[5];

— офіційна підтримка фреймворку;

— краща видимість додатку в Google Play.

Під кросплатформними розуміються засоби, за допомогою яких створюються програми, які сумісні з кількома операційними системами, і тому можуть працювати на будь-якому смартфоні чи планшеті[6].

Після огляду всіх існуючих кросплатформних рішень було виділено наступні переваги:

— швидкість написання з нуля. За умови вибору правильного технологічного стеку і ретельно спланованого проекту, між платформами може бути перевикористано до 80% вихідної кодової бази;

— економічна ефективність. Розробка одного додатка для двох платформ дешевша за розробку двох нативних рішень;

— випуск оновлень. У світі, де успішні видавці додатків розгортають оновлення до 4 разів на місяць, витрати на технічне обслуговування можуть забрати велику частину всього прибутку додатків — і саме там виникає міжплатформна розробка.

Але подальші дослідження показали, що незважаючи на те що наявні переваги кросплатформних засобів є досить вагомими, вони були знівельовані наступними недоліками:

— проблеми із швидкодією. Обчислювальна потужність смартфонів відносно невелика. З іншого боку, відтворення важких компонентів користувальницького інтерфейсу HTML5 / CSS[7] займає багато ресурсів GPU / CPU, і може збільшити час відгуку додатка[8];

— проблеми з UX. Задоволення вимог UX для обох платформ Android, iOS може бути складним завданням. Apple особливо відома своїми інструкціями щодо побудови інтерфейсу користувача і не допускає до маркету додатки, що являють собою веб-сайти, загорнуті у нативний контейнер;

— нестабільність та застарілість деяких кросплатформних рішень. Більшість кросплатформних засобів розробляються компаніями що не є розробниками нативного SDK[9], і як результат, часто не встигають реалізувати функціонал, подібний до того що виходить у світі нативної розробки;

— повнота та якість документації. Нативні рішення більш популярні у світі, краще задокументовані а також краще обговорюються в літературі та технічних сайтах, присвячених розробці ПЗ.

В результаті аналізу був обраний перший варіант як більш стабільний та документований, що підвищує швидкість імплементації функціоналу, а також звичний для кінцевого споживача через переваги в UX та швидкодії.

Висновки до розділу 2

Існуючі програмні засоби не є повними замінниками проектованої системи, а тільки можуть її частково доповнювати. З точки зору методів розробки, то аналіз показав, що використання нативних засобів є найоптимальнішим у даній ситуації.

3 МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Аналізуючи поставлену задачу та методи її вирішення, було вирішено розроблювати додаток, який є гнучким, сучасним та легким у використанні.

3.1 Середовище розробки Android Studio/IntelliJ IDEA

Розробку застосунків для Android можна вести мовою Java та Kotlin. Офіційним середовищем розробки є Android Studio, представлене компанією Google в 2013. Android Studio[10] зображена на рисунку 3.19.

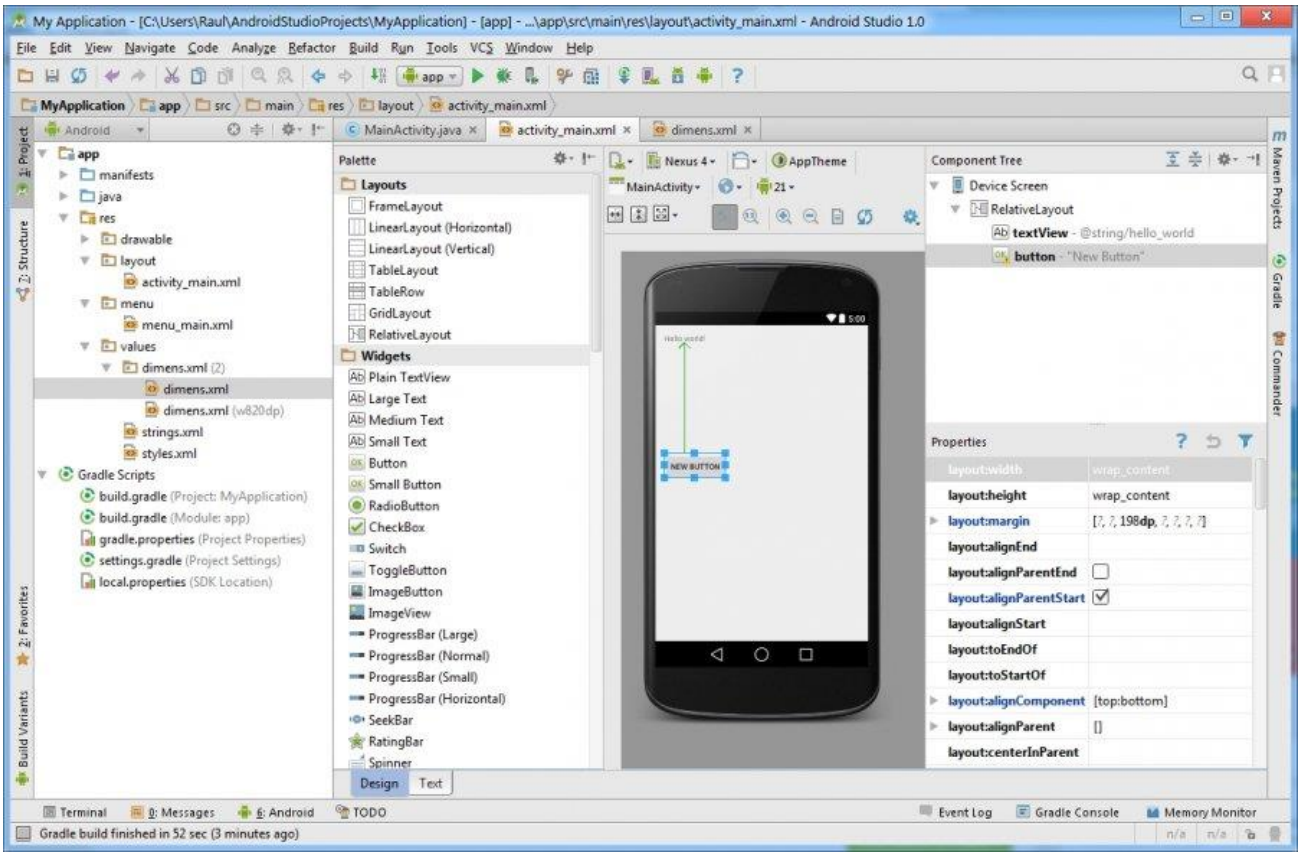

Рисунок 3.1 – Android Studio

Крім цього існує плагін для Eclipse — «Android Development Tools» (ADT)[11], призначений для Eclipse версій 3.3-3.7. Для IntelliJ IDEA також існує плагін, який

полегшує розробку Android-застосунків.. Для середовища розробки NetBeans розроблено плагін, який починаючи з версії Netbeans 7.0 перестав бути експериментальним. Крім того існує Motodev Studio for Android, що являє собою комплексне середовище розробки, засноване на базі Eclipse і дозволяє працювати безпосередньо з Google SDK. Крім того в 2009 році на застосунок до ADT був опублікований Android Native Development Kit (NDK)[12], пакет інструментаріїв і бібліотек дозволяє вести розробку застосунків мовою C/C++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості.

Основним середовищем розробки серверної частини на Java/Kotlin є IntelliJ IDEA[13], що розроблена компанією JetBrains і має єдину платформу із Android Studio IDE.

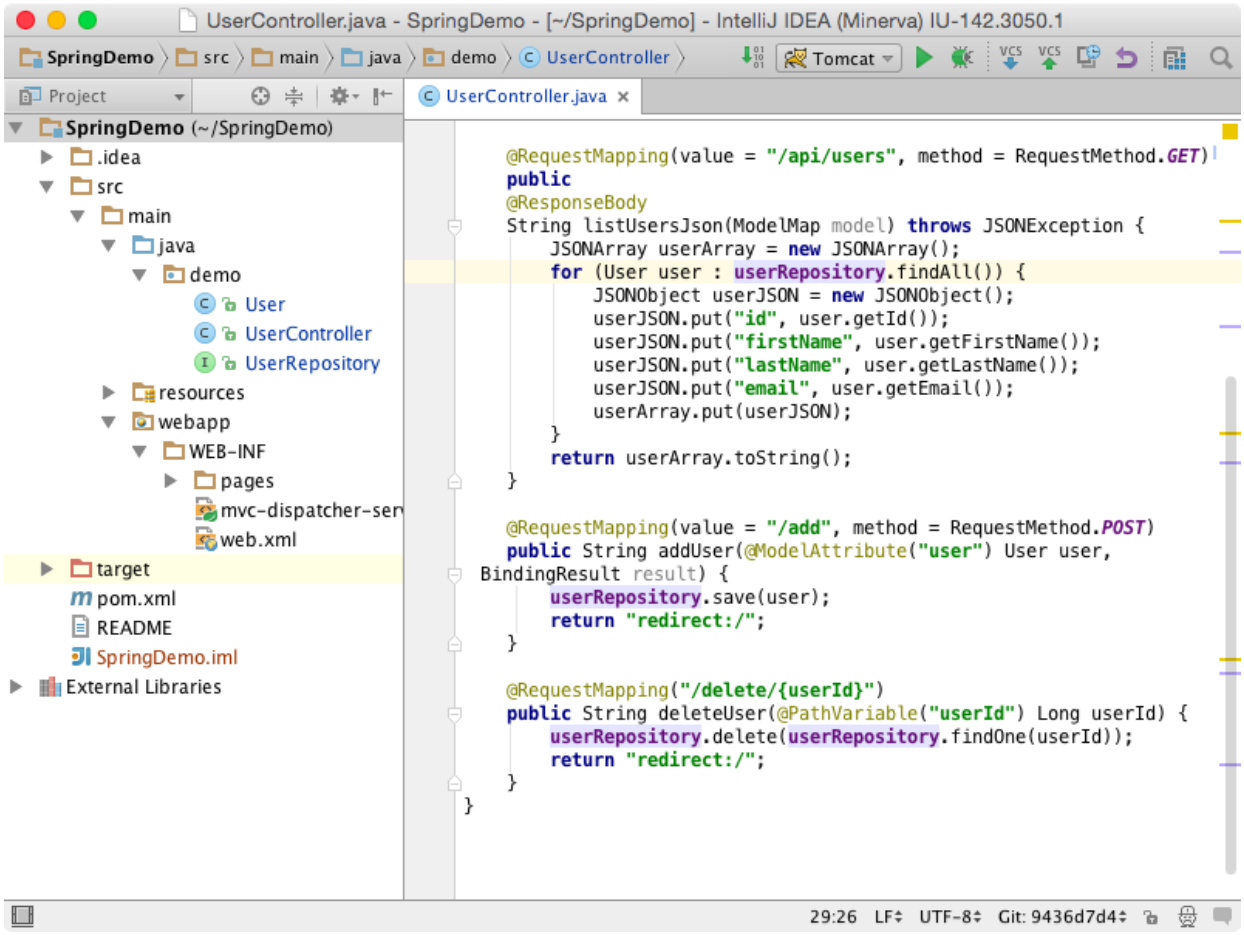

Рисунок 3.2 – IntelliJ IDEA

Дане середовище існує у двох модифікаціях: безкоштовна та повна. Для комерційної розробки серверних додатків необхідно використовувати саме повну

версію середовища. У ньому наявні додаткові плагіни для роботи із базами даних та проектування архітектури додатку, а також підтримка різних фреймворки, серед яких є і Spring Framework.

InjelliJ IDEA постійно розвивається, і функції, що в ній з'являються, пізніше переходять і до Android Studio.

3.2 Особливості мови Kotlin

Мова Kotlin[14] — , що працює поверх і розробляється компанією . Позиціонується розробниками як об'єктно-орієнтована мова промислового рівня, а також як мова, яка зможе замінити Java[15]. При цьому мова повністю сумісна з Java, що дозволяє розробникам поступово перейти з Java на Kotlin. Зокрема, в Android мова вбудовується за допомогою Gradle[16], що дозволяє для існуючого Android-додатки впроваджувати нові функції на Kotlin без переписування програми цілком.

Автори ставили перед собою ціль створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala[17]. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка мови IDE.

Одним з основних місць застосування Котліна є розробка під Android. Платформа на деякий час застрягла на Java 7 (деякі сучасні мовні функції стали доступними завдяки використанню Retrolambda або інструментарію Jack), а Котлін вносить багато поліпшень для програмістів, таких як null-безпечність, функції розширення та інфіксна нотація. Завдяки повній сумісності з Java та підтримкою IDE (Android Studio), він покликаний покращити читання коду, полегшити використання класів Android SDK та загально прискорити розробку.

Котлін був оголошений офіційною мовою розробки Android в Google I/O 2017. Це стало третьою мовою, повністю підтримуваною для Android, крім Java та C ++.

Згідно з блогом Jetbrains, Kotlin використовується у Amazon Web Services, Pinterest, Coursera, Netflix, Uber, Square, Trello, Basecampта ін. Corda, компанія, що обслуговує відомі банки (такі як Goldman Sachs, Wells Fargo, J.P. Morgan, Deutsche

Bank, UBS, HSBC, BNP Paribas, Société Générale), має кодову базу, що на більше ніж 90% складаєтсья з Kotlin.

Згідно з Google, Kotlin також використовується відомими програмістами з таких компаній як Expedia, Flipboard, Square[18], та інших.

До основних переваг мови Kotlin перед мовою Java є:

— null-safety на етапі компіляції[19]. Система типів в мові Kotlin націлена на те, щоб викорінити небезпеку звернення до null значень, більш відому як "Помилка на мільйон"[20]. Найпоширенішим підводним каменем багатьох мов програмування, в тому числі Java, є спроба зробити доступ до null значень. Це призводить до помилки. В Java така помилка називається NullPointerException[21];

— властивості замість гетерів та сетерів. Класи в Kotlin можуть мати властивості: змінювані (mutable)[22] і незмінні (read-only) — var і val відповідно. Синтаксис оголошення констант має дві відмінності від синтаксису оголошення змінюваних змінних: по-перше, оголошення починається з ключового слова val замість var, а по-друге, оголошення сетера заборонено. Класи в Kotlin не можуть мати полів. Тобто змінні, які ви оголошуєте всередині класу тільки виглядають і поводяться як поля з Java, хоча насправді є властивостями, тому що для них неявно реалізуються методи get і set[23]. А сама змінна, в якій знаходиться значення властивості, називається backing field[24];

— широкий спектр методів для роботи з колекціями. На відміну від багатьох мов, Kotlin розрізняє змінювані і незмінні колекції (списки, множини, асоціативні списки і т.д.). Точний контроль над тим, коли саме колекції можуть бути змінені, корисний для усунення багів і розробки хорошого API[25]. Важливо розуміти різницю між read-only поданням змінною колекції, і фактично незмінною колекцією. Їх легко створити, але ось система типів не виражає відмінність між ними, тому стежити за цим повинні ви (якщо це необхідно);

— зручність об’явлення класів. Клас в Kotlin може мати первинний конструктор (primary constructor) і один або більше вторинних конструкторів (secondary constructors). Первинний конструктор є частиною заголовка класу, його оголошення йде відразу після імені класу. У класах також можуть бути оголошені

додаткові конструктори (secondary constructors), перед якими використовується ключове слово constructor. Властивості, оголошені в первинному конструкторі, можуть бути змінювані (var) і незмінні (val). Якщо у конструктора є анотації або модифікатори видимості, ключове слово constructor обов'язково, і модифікатори використовуються перед ним;

— значення параметрів за замовченням. Параметри функції можуть мати значення за замовчуванням, які використовуються в разі, якщо аргумент функції не вказано при її виклику. Це дозволяє знизити рівень перевантаженості коду в порівнянні з іншими мовами. Перевизначені методи завжди використовують ті ж самі значення за замовчуванням, що і їх базові методи. При перевизначенні методів зі значеннями за замовчуванням ці параметри повинні бути опущені;

— делегати властивостей та делеговані класи. Стандартна бібліотека Kotlin надає кілька корисних видів делегатів: ледачі властивості[26] lazy() це функція, яка приймає лямбда і повертає екземпляр класу Lazy<T>, який служить делегатом для реалізації ледачого властивості: перший виклик get() запускає лямбда-вираз, передане lazy() як аргумент, і запам'ятовує отримане значення, а наступні виклики просто повертають обчислене значення. Функція Delegates.observable()[27] приймає два аргументи: початкове значення властивості і обробник (лямбда), який викликається при зміні властивості. У обробника три параметра: опис властивості, яке змінюється, старе значення і нове значення.;

— extension-функції[28]. Аналогічно до таких мов програмування, як C#[29] і Gosu[30], Kotlin дозволяє розширювати клас шляхом додавання нового функціоналу. Чи не успадковуємо від такого класу і не використовуючи патерн "Декоратор"[31]. Це реалізовано за допомогою спеціальних виразів, які називаються розширення. Kotlin підтримує функції-розширення і властивості-розширення. Розширення насправді не проводять ніяких модифікацій з класами, які вони розширюють. Оголошуючи розширення, ви створюєте нову функцію, а не новий член класу. Зі сторони Java також можливо викликати методи розширення. Усі методи розширення представляються у вигляді статичних методів, а клас, з якого вони викликаються, стає першим параметром статичного метода.

3.3 Інструментарій для веб-застосунків Spring Framework

Spring Framework[32] — це програмний каркас з відкритим кодом та контейнери інверсії управління для платформи Java.

Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але на платформі є розширення для створення веб-додатків Java EE[33]. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування. Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean.

Spring Framework забезпечує комплексну модель розробки і конфігурації для сучасних бізнес-додатків на Java — на будь-яких платформах. Ключовий елемент Spring — підтримка інфраструктури на рівні програми: основна увага приділяється "водопроводу" бізнес-додатків, тому розробники можуть зосередитися на бізнес-логіці без зайвих налаштувань в залежності від середовища виконання.

Spring Framework складається з кількох модулів, які надають широкий спектр послуг:

— контейнер Інверсії управління[34]: Конфігурація компонентів додатків і управління життєвим циклом об'єктів Java, здійснюється головним чином через Інверсію управління;

— аспектно-орієнтоване програмування[35]: дозволяє реалізувати наскрізні процедури;

— доступ до даних: робота з реляційною системою управління базами даних на платформі Java з використанням JDBC[36] і об'єктно-реляційні відображення та інструментів з NoSQL[37] баз даних;

— управління транзакціями: об'єднує кілька API, управління транзакціями та координує операції для Java-об'єктів;

— Модель-Вигляд-Контроллер (Model-View-Controller)[38]: програмний каркас на основі HTTP сервлета[39], що забезпечує створення веб-додатків і веб-служб RESTful;

— аутентифікація і авторизація: налаштовувані процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою підпроекту Spring Security (колишня система безпеки AcerI для Spring);

— віддалене керування: конфігураційний вплив і управління Java-об'єктами для місцевої (локальної) або віддаленої конфігурації через JMX;

— тестування: підтримка класів для написання юніт-тестів та інтеграційних тестів.

Структура Spring Framework зображено на рисунку 3.1.

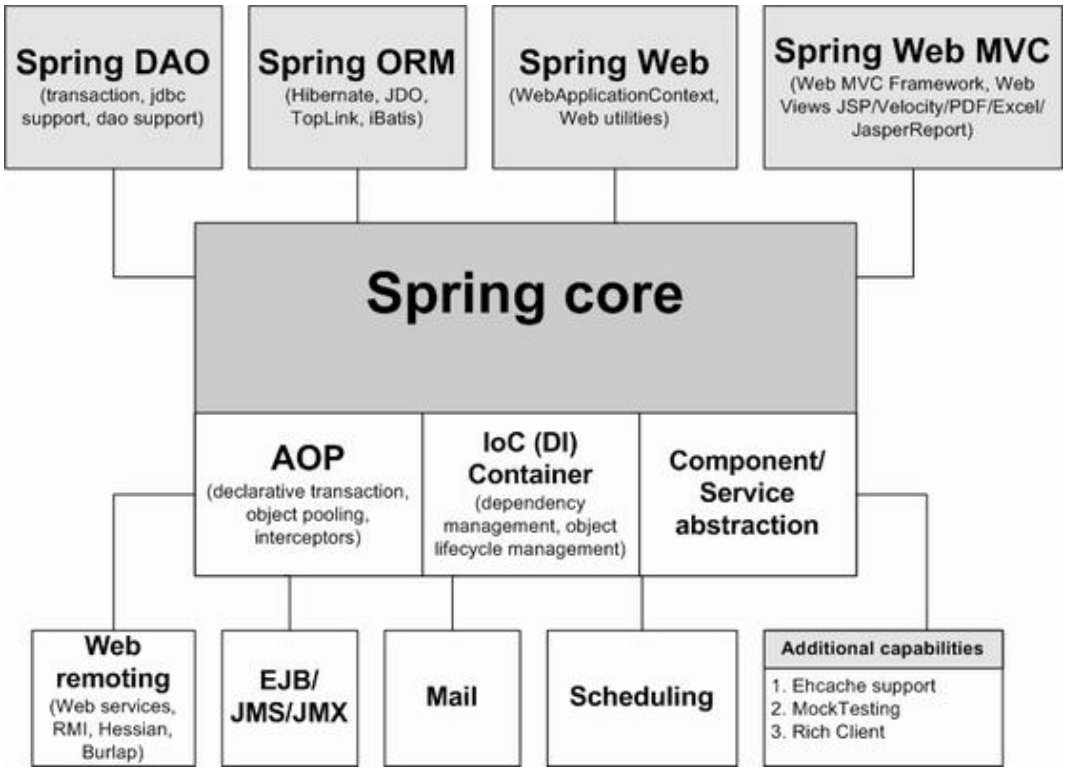

Рисунок 3.1 — Структура фреймворку Spring

3.4 ORM Hibernate

Hibernate[40] — засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об'єктно-орієнтованою моделлю даних і традиційною реляційною базою даних реляційною базою даних.

Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати Hibernate як при розробці з нуля, так і для вже існуючої бази даних.

Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Hibernate забезпечує прозору підтримку збереження даних, тобто їхньої персистентності (persistence) для звичайних Java-об'єктів; єдина сувора вимога до класу, що зберігається — конструктор за замовчанням (Для коректної поведінки у деяких застосуваннях потрібно приділити особливу увагу до методів equals() і hashCode()).

Mapping Java класів з таблицями бази даних здійснюється за допомогою конфігураційних XML[41] файлів або Java анотацій. При використанні файлу XML, Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання (persistent). У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості з організації відношення між класами «один-до-багатьох» і «багато-до-багатьох». На додаток до управління зв'язками між об'єктами, Hibernate також може керувати рефлексивними асоціаціями, де об'єкт має зв'язок «один-до-багатьох» з іншими примірниками свого власного типу даних.

Hibernate підтримує відображення користувацьких типів значень. Це робить можливим такі сценарії:

— перевизначення типу за замовчуванням SQL, який Hibernate вибирає при відображенні стовпчика властивості;

— mapping перераховуваного типу Java до колонок БД, так ніби вони є звичайними властивостями;

— mapping однієї властивості в декілька колонок.

На рисунку 3.2 зображеано структуру взаємодії ORM Hibernate із Java-додатком. У структурі фреймворку є такі поняття як сесія, транзакція, крітерій пошуку, запит, та конфігурація. Для доступу до бази даних використовуються такі більш низькорівневі структури як JDBC.

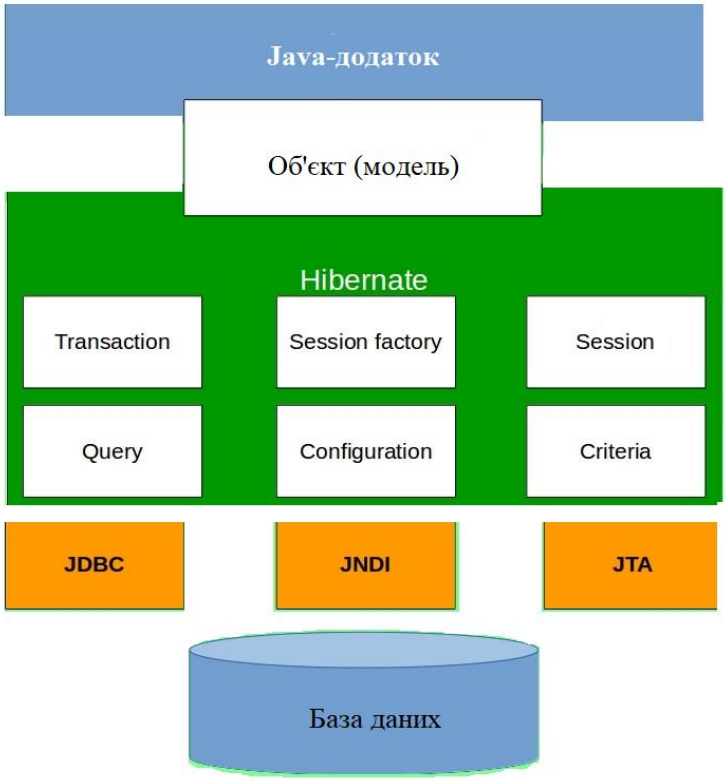

Рисунок 3.2 — Структура Hibernate ORM

З іншого боку бази даних, на стороні додатку, відбувається взаємодія із звичайними моделями, як показано на рисунку 3.2.

3.5 Бібліотека для обробки потоків даних Rx Java

RxJava[42] реалізує концепти реактивного та функціонального програмування.

З одного боку, функціональне програмування — це процес побудови програмного забезпечення шляхом складання чистих функцій, уникнення спільного стану, змінних даних та побічних ефектів.

З іншого боку, реактивне програмування — це асинхронна парадигма програмування, що стосується потоків даних та поширення змін.

Спільне функціональне реактивне програмування являє собою комбінацію функціональних та реактивних методів, які можуть представляти елегантний підхід до керування подіями — із значеннями, які змінюються з часом і де споживач реагує на дані, як це відбувається.

Ця технологія об'єднує різні реалізації її основних принципів, деякі автори придумали документ, який визначає загальний словник для опису нового типу програм — маніфест.

Реактивний маніфест[43] — це онлайновий документ, який встановлює високі стандарти для програм у галузі розробки програмного забезпечення. Простіше кажучи, реактивними системами є:

— реакційні — системи повинні відповісти своєчасно;

— контрольовані повідомленнями — системам слід використовувати асинхронні повідомлення між компонентами для забезпечення взаємодії;

— еластичні — системи повинні залишатися чутливими під високим навантаженням або коли деякі компоненти не працюють.

Ці концепти реалізовано у бібліотеці за допомогою сутностей двох типів — потоку даних та підписника на потік.

Класи потоків містять великий набір методив для перетворення даних у потоці або для перетворення одних потоків на інші (map, flatmap, filter та ін.)

Класи підписників вміють реагувати на порції нових даних у потоці, на виникнення помилки чи завершення роботи потоку.

Дана бібліотека допомогає виконувати асинхронні операції, особливо коли ці операції необхідно виконувати за певними правилами (послідовно чи паралельно, або за певної умови).

До основних операцій rx java відносяться:

— map — перетворення об’єкту одного типу в об’єкт іншого типу;

— flatMap — перетворення об’єкту одного типу в потік об’єктів іншого типу;

— filter — фільтрація об’єктів в потоку за певним правилом;

— merge — поєднання кількох паралельних потоків даних в один;

— zip — попарне поєднання даних з кількох потоків.

Також до основних операцій створення потоків даних належать:

— just – перетворює об’єкт в потік;

— fromIterable – перетворює колекцію в потік;

— fromCallable – перетворює результат функції в потік.

3.6 Організація клієнт-серверної взаємодії зі сторони мобільного додатку

Для виконання запитів до серверу було обрано веб-клієнт OkHttp та його зручну обгортку Retrofit[44].

Веб-клієнт OkHttp зарекомендував себе як швидкий, універсальний та конфігурований веб-клієнт, що дозволяє робити запити як асинхронно, так і синхронно. Також зручною особливістю даного веб-клієнту є можливість логувати усі запити та відповіді, тип самим значно полегшується процес відладки застосунків. Даний веб-клієнт може бути використаним у будь-якій програмі на мові Java, але популярністю від користується саме в контексті програмування під Android. Обгортка веб-клієнту Retrofit дозволяє значно полегшити конфігурування запитів веб-клієнту та обробки відповідей, що надходять з веб-клієнту, завдяки набору анотацій, згідно з якими на етапі компіляції генерується код конфігурування запитів веб-клієнту, тим сам код стає прозорішим, а програміст не витрачає час на написання подібного шаблонного коду.

До основних анотацій фреймворку відносяться:

— POST, GET, PUT, DELETE[45] — методи запитів. У якості параметрів приймають абсолютний або відносний шлях до апі сервера;

— Query — url-параметр запиту. У якості параметру приймає ключ параметру у запиті;

— Path — параметр шляху запиту;

— Body — json-тіло запиту;

— Headers — додаткові хедери запиту.

Дана бібліотека також має можливість автоматично переводити об’єкти в json-об’єкти (або xml) і навпаки, використовуючи адаптери інших бібліотек, щоспрямовані на серіалізацію та десеріалізацію об’єктів у json/xml. На даний момент бібліотека має адаптери на наступні сериализатори:

— Gson;

— Jackson;

— Moshi;

— Protobuf;

— Wire;

— Simple XML.

Також є можливість писати свої адаптери та парсери. Одним з розповсюджених прикладів використання кастомних адаптерів є обробка полів в яких зберігається long-змінна, що відповідає за час у секундах/мілісекундах. Подібні поля на стороні клієнта зручніше зберігати у виді об’єктів класу Date.

Висновки до розділу 3

Даний проект використовує сучасні технології як серверної, так і мобільної розробки. Усі перераховані та описані вище технологій на даний момент є провідними, а більшість з них є необхідними у будь-якому сучасному додатку. Ці технології полегшують написання та підтримку проектів, а їх високорівневий інтерфейс є самодокументуючим для коду програм, що дозволяє зекономити робочі години під час вивчення структури додатку розробником.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Даний сервіс складається з двох модулів — сервер для зберігання та обробки даних користувачів і мобільний додаток, що дозволяє оперувати даними, що знаходяться на сервері (створювати, редагувати та видаляти дані, проводити пошук та відтворення через користувацький інтерфейс).

4.1 Опис функціональності системи

Під час аналізу предметної області будо виділено акторів системи а також операції, що вони можуть виконувати. На рисунку 4.1 зображено діаграму прецендентів[46] додатку.

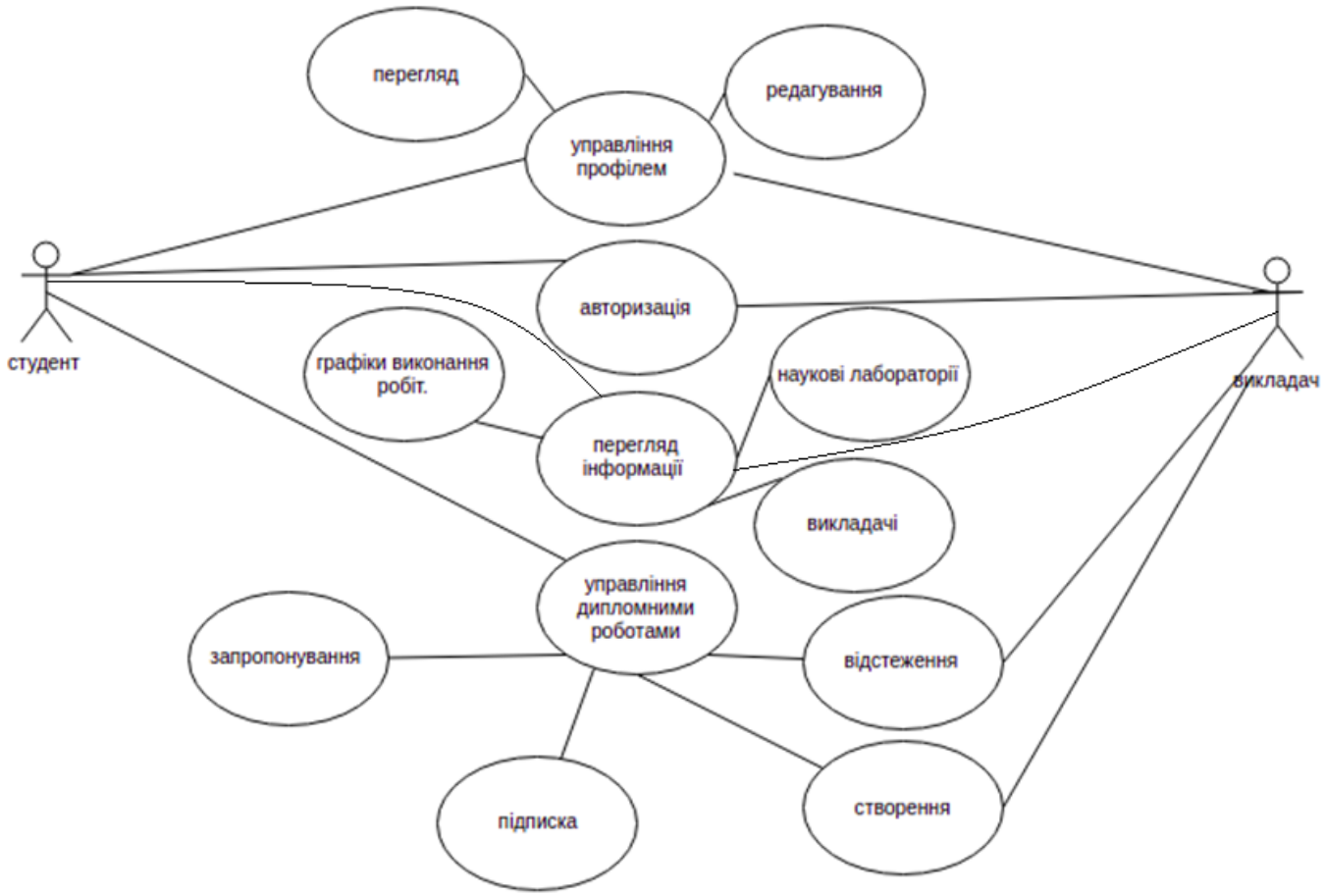

Рисунок 4.1 — Діаграма прецендентів системи

Як видно з діаграми, було виділено двох акторів системи — студента та викладача. Майже всі прецендети є спільними для обох акторів. Серед цих прецендентів є:

— авторизація;

— управління профілем;

— перегляд інформації;

— управління дипломними роботами.

Різниця між акторами становить в тому що викладач, на відміну від студенту, не має можливості самостійно підписуватися на створені студентом дипломні роботи з причини того що в застосунку немає можливості переглядати список студентів, на відміну від перегляду списку викладачів, що наявний в додатку.

З тих сами причин викладач не має можливості запропонувати свою роботу студенту. Таким чином, задача викладача полягає саме в створенні та оголошені списку своїх тем.

4.2 Опис моделі даних системи

Вхідними даними є дії користувача, що направлені на взаємодію із сутностями (дипломна робота, лабораторія, викладач та ін.). Дані зберігаються на віддаленому сервері у реляційній базі даних, а задання параметрів — завдання клієнтського додатку.

Вихідна інформація — сутності, що було створено на віддаленому сервері, запит і відображення яких відбуваються у клієнтському додатку.

Одним з перших етапів створення програмного продукту є виділення сутностей а також встановлення зв’язків між ними.

До основних сутностей відносяться:

— студент;

— викладач;

— лабораторія;

— напрям лабораторії;

— дипломна робота.

Для визначення зв’язків необхідно виділити обов’язки сутностей (юзекейси).

Було виділено наступні властивості:

1. студент і викладач можуть бути авторами теми для дипломної роботи;

2. і студент, і викладач мають можливість реєструватися в системі та редагувати дані про себе;

3. дипломна тема має кілька станів. Її можна затвердити чи відхилити, а також запропонувати кільком студентам чи викладачам тощо;

4. студенти можуть пропонувати свої теми одразу кільком викладачам, але тільки тим, у яких ще є місце на наукову роботу.

Як результат, було спроектовано схему бази даних[47], що зображено на рисунку 4.2:

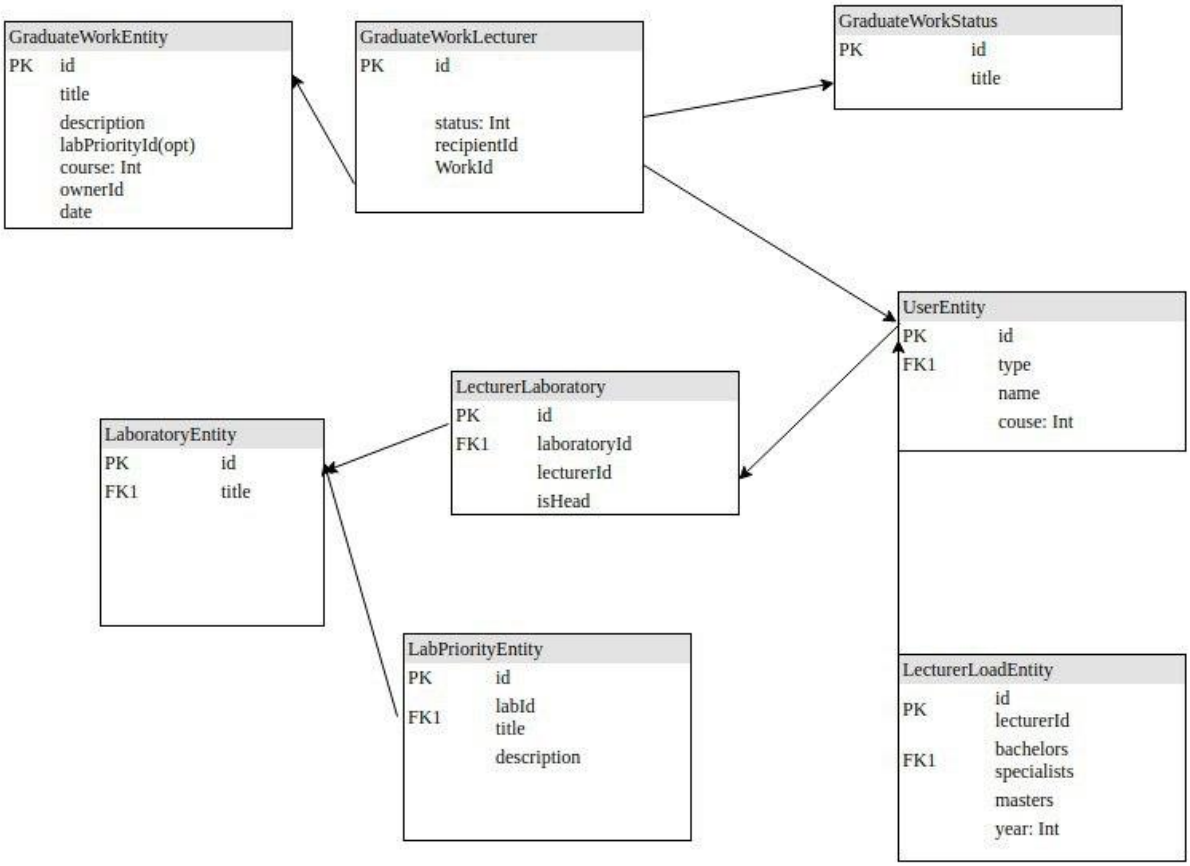

Рисунок 4.2 — Схема бази даних

4.3 Опис підходу до архітектури системи

Було вивчено найпоширеніші архітектурні патерни що використовуються в розробці під Android. Серед них: MVC, MVP, MVVM[48] для реалізації UX/UI частини додатку, а також Clean Architecture[49] як приклад комплексної архітектури всього додатку, а також патерни Observer, Proxy та Façade[50] і принципи SOLID[51].

Як результат, було запропоновано шаблонну архітектуру, що лягла в основу проекту, і є результатом комбінації вище перерахованих патернів та принципів.

Ця архітектура відповідає наступним характеристикам:

— незалежність від фреймворків;

— тестованість;

— незалежність від UI;

— незалежність від БД;

— незалежність від будь-якої зовнішньої служби.

Для забезпечення максимальної стрункості та прозорості архітектури було прийнято рішення розділити логіку на окремі рівні обв’язків. До цих рівнів відносяться:

— сутності — бізнес-логіка додатку;

— методи використання (посередники) — методи, що організують потік даних в сутності та з них;

— інтерфейси-адаптери — набір адаптерів, що конвертують дані одного типу в дані іншого типу;

— фреймворки та драйвери — місце скопичення деталей — UI, інструменти, фреймворки, бази даних та ін. Тільки цей рівень логіки має прив’язку до контексту, в якому розробляється додаток, тобто містить реалізації із конкретною прив’язкою до зовнішнього коду.

На рисунку 4.3 зображено схему прозорої стрункої архітектури. За допомогою вкладеності показана залежність одних рівнів від інших. Вона йде ззовні всередину, тобто рівень фреймворків залежить від адаптерів, і т.д, і навпаки, рівень сутностей є

базовим і не залежить ні від чого. На сутностях побудовані посередники, які більш ні від чого не залежать, і т.д.

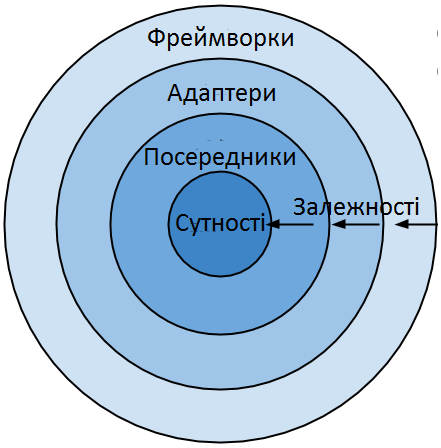

Рисунок 4.3 — Схема рівнів струнокої архітектури

Також вдосконаленням архітектури в даній магістерській роботі є реалізація взаємодії між модулями через потоки даних, що являють собою модифікацію шаблону Observer, де постачальник подій не знає про слухачів, а між постачальником і слухачем на різних рівнях логіки вбудовуються адаптери, що модифікують дані у вид, прийнятний для обробки слухачем.

Абстрагування між шарами логіки у контексті типів даних та низькорівневих операцій досягається за допомогою потоків даних реактивного програмування. На рисунку 4.4 зображено деякі основні операції з потоками даних.

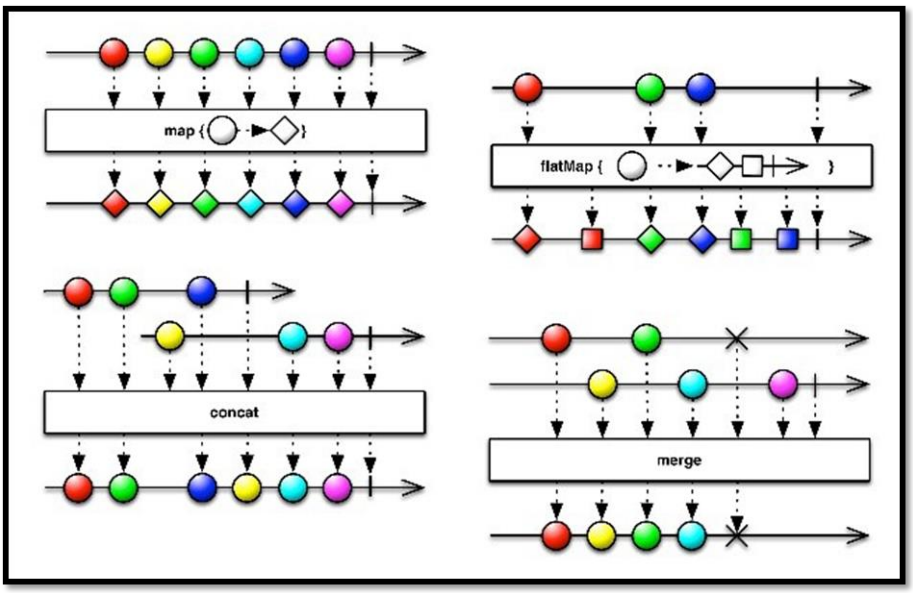

Рисунок 4.4 — Основні операції потоків даних

У якості основи шаблону на рівні представлення було обрано основу патерну MVP та вдосконалено його для кращої сумісності із фреймворком Android.

Шаблон MVP (Model-View-Presenter) — це архітектурний шаблон проектування, головною ідеєю якого є відділити логіку застосунку від представлення (рисунок 4.5). Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності. Використовується для побудови користувацьких інтерфейсів. Принципом MVP є розділення програмної реалізації системи на три головні компоненти: M — Model (Модель), V — View (Представлення), P — Presenter (Представник), таким чином, що редагування будь-якого компонента може відбуватися незалежно.

Модель — містить знання про предметну область, дані та правила до цих даних, але нічого не знає про контролери та представлення. У моделі відбувається бізнес-логіка роботи застосунку. Модель надає контролеру дані які запрошує користувач, або інша система.

Представлення — надає можливість по-різному відображати дані отримані будь-яким способом від моделі, може містити в собі логіку. Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення.

Представник — реалізує взаємодію між моделлю і представленням.

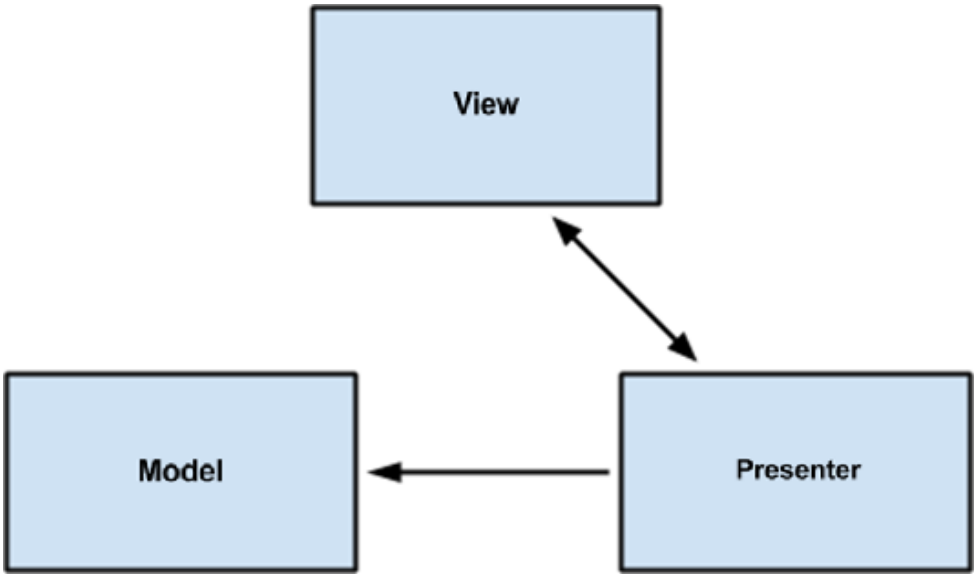

Рисунок 4.5 — Схема роботи MVP-шаблону

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого поділу підвищується можливість повторного використання та гнучкість системи.

У Шаблоні MVP представники виступають в ролі адаптерів згідно з системою рівнів стрункої архітектури, реалізації представлень та моделей відносяться до рівня фреймворків, а їх інтерфейс — до рівня адаптерів.

Вдосконалення вище описаного патерну в контексті шаблонної архітектури є своєрідне встановлення обов’язків між представленням та презентером а також додавання обробки android-специфічної поведінки — підтримки поворотів екрану, із збереженням прогресу асинхронних операцій та кешуванням результатів цих операцій із ціллю відображення актуальних даних при будь-яких модіфікаціях стану додатку.

Принцип SOLID інверсія управління допомагає підвищити модульність у програмі, тим самим зміна та додавання нового функціоналу здійснюється набагато легше. Суть даного принципу зазвичай викладається наступним чином:

— модулі верхнього рівня не повинні залежати від модулів нижнього рівня. І ті, і інші повинні залежати від абстракцій;

— абстракції не повинні залежати від деталей. Деталі повинні залежати від абстракцій.

Даний принцип є актуальним для будь-якої об’єктно-орієнтованої мови та області застосування, але в контексті шаблонної архітектури його було вдосконалено шляхом додавання областей видимості та життя об’єктів.

Було виділено наступні області видимості (рисунок 4.6):

— сінглтон рівня існування додатку в оперативній пам’яті;

— сінглтон рівня авторизованого користувача;

— сінглтон рівня життя екрану із переживанням повороту екрану.

Дані рівні життя реалізуються за допомогою кастомних анотацій та встановлення рівня зв’язків між модулями, що генерують сінглтони, та їх інтерфейсами – компонентами, через які і відбувається введення залежностей.

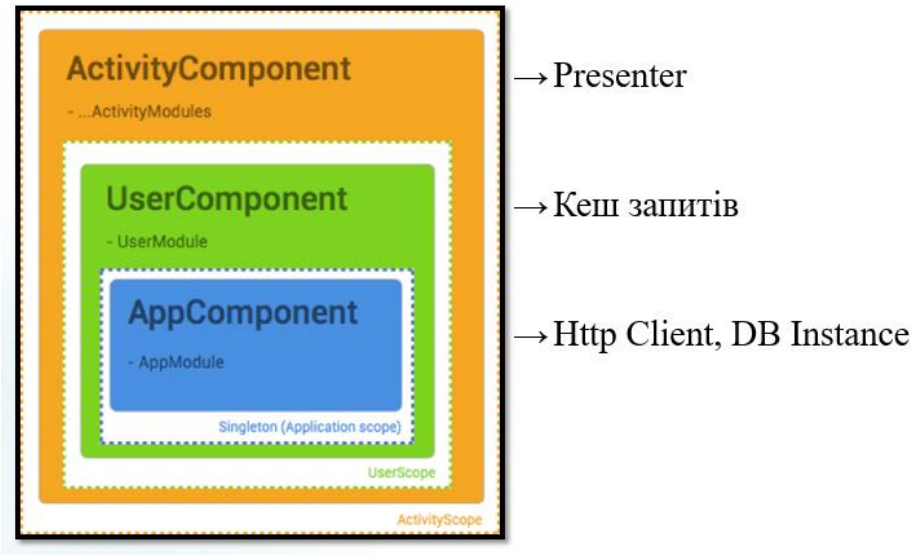

Рисунок 4.6 – Схема областей видимості сінглтонів

На рисунку 4.7 зображено діаграму послідовностей[52], що демонструє типовий flow викликів функцій під час взаємодії із користувачем. Дана діаграма описує ситуацію коли користувач відкриває екран та виконує операцію взаємодії з інтерфейсом, після чого додаток виконує певні операції взаємодії із базою/сервером, і результат операції відображає на екрані.

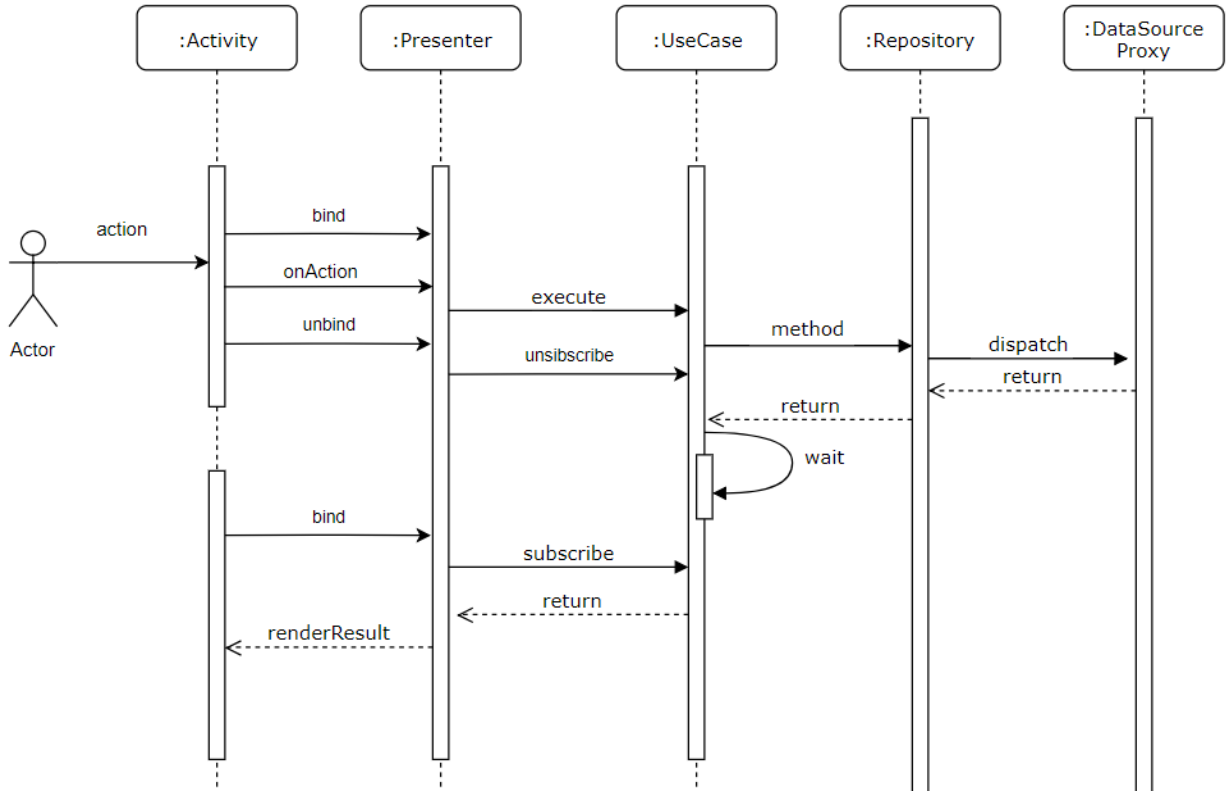

Рисунок 4.7 — Діаграма послідовностей

4.4 Розробка модуля роботи з сервером

Даний модуль є обслуговуючим для усіх модулів клієнтського додатку і містить в собі набір методів для обміну даними із віддаленим сервером.

Сервер сервісу для організації роботи кафедри — написаний спеціально для даного застосунку і містить інтерфейс для взаємодії клієнта із сутностями баз даних та методи обробки даних, що надходять від клієнта. Даний сервер містить близько 20 методів Rest api[53], і майже кожний з них має своє відображення у клієнтському додатку (наявні також методи для майбутнього розширення клієнтського додатку).

Модуль роботи з сервером представлений двома класами: “RetrofitApi”, що містить у собі анотований описовий код-відображення методів api та “RestApi”, що є обгорткою навколо “RetrofitApi”. Із методами api додаток спілкується тільки через “RestApi”, що за своєю суттю відноситься до третього рівня стрункої архітектури — до рівня адаптерів. “RetrofitApi”, у свою чергу, відноситься до четвертого рівня — рівня фреймворків.

На даному етапі розробки додатку для зв’язку із сервером зручно було використовувати саме фреймворк Retrofit, переваги якого було описано раніше, але в майбутньому може виникнути необхідність замінити даний фреймворк іншим (причинами цього можуть стати зупинка підтримки фреймворку розробниками, поява більш ефективного фреймворку, зміна прав користання та інші), а так як взаємодія із сервером відбувається за допомогою адаптера “RestApi”, що працює із сутностями бізнес-логіки додатку з одного боку, та фреймворками з іншого, то правки доведеться вносити тільки до цього класу, і модулі, що взаємодіють із сервером через дану обгортку, не будуть змінені жодним чином. “RestApi” також відповідальний за стиснення зображень, тобто займається конвертуванням шляхів до зображень на постійному сховищі в стиснені файли, що готові для відправки на сервер, при чому сама логіка роботи зі сховищем та стиснення зображень винесена в спеціальні обслуговуючі класи “FilesUtils” та “ImageUtils” відповідно. Таким чином, інші модулі не знають про перетворення файлів, а низькорівневий код винесений в спеціальні обслуговуючі класи.

Попередньо було згадано деяку кількість посередників. Кожний з них відноситься до другого рівня архітектури і взаємодіє із сутностями бізнес-логіки через спеціальний репозиторій, що реалізовує інтерфейс “Repository”. Реалізація даного інтерфейсу звертається до методів класу “RestApi”, що, в свою чергу, звертається до методів “RetrofitApi”.

4.5 Розробка модуля роботи із обліковим записом

Модуль роботи із обліковим записом зберігає в собі логіку, що відповідає за загальні для всіх користувачів функції. До цього модуля не відносяться функції, специфічні для даної системи. Зазвичай подібний модуль міститься у кожному клієнтському додатку і відповідає за налаштування сутності користувача. Цей модуль є необхідною частиною всією системи. Під час роботи із даним модулем нема розділення на користувача-водія та користувача-пасажира. Даний модуль відповідає загалом за створення, редагування та оцінювання користувачів і поділяється на дві частини:

— модуль авторизації;

— модуль управління профілем.

4.5.1 Модуль авторизації

Модуль авторизації забезпечує функціонал, що створювати та автентифікувати користувачів, використовуючи адресу електронної пошти та інші дані. Загалом даний модуль включає в себе такий функціонал:

— авторизація;

— валідація локально збережених даних;

— збереження та відновлення даних облікового запису.

Даний модуль включає в себе три екрани:

— екран валідації — відповідає за перевірку локально збережених даних облікового запису. Результатом роботи даного екрану є перехід на екран облікового запису конкретного користувача, або перехід на екран авторизації;

— екран авторизації — пропонує користувачу ввести дані для входу в обліковий запис (email та пароль), після чого виконує запит до сервера, та у випадку позитивної відповіді від сервера зберігає дані облікового запису та здійснює перехід на екран облікового запису авторизованого користувача. Даний екран також дозволяє здійснити перехід на екран реєстрації;

Під час авторизації може виникнути багато різноманітних помилок. Частину цих помилок можна перевірити на стороні клієнта без звернення до сервера (незаповнені або невірно заповнені поля, відсутність інтернет-з'єднання), а іншу частину помилок надсилає сервер. Для обробки помилок, пов'язаних із невірним заповненням полів, створено клас-виняток "FieldDataException", а обробка полів здійснюється у методах класу "FieldUtils". До бізнес-логіки даних даного модулю відноситься посередник: "LoginUseCase", що містть логіку взаємодії із методами авторизації та реєстрації репозиторія "Repository"[54].

4.5.2 Модуль управління профілем

Даний модуль забезпечує функціонал, що відповідає за перегляд та модифікування профілю і містить три екрани:

— екран перегляду профілю — дозволяє переглядати профіль будь-якого користувача. Даний екран у вхідних параметрах приймає сутність "User", і в залежності від того, чи дана сутність відображає користувача даного облікового запису, екран наділяється додатковим функціоналом. Зокрема, при роботі з сервером може виникнути необхідність отримати інформацію про певного користувача, а отже дана інформація у повному обсязі зберігається у JSON-відповіді сервера [55], тому екран профілю матиме всю інформацію для відображення у своїх вхідних параметрах. У свою чергу, при відображенні даних користувача авторизованого облікового запису сутність "User" поступає до екрану профілю не у повному обсязі через те, що локально зберігаються тільки ті дані, що користувач облікового запису може редагувати тільки самостійно, а отже вони не втрачають свою актуальність. Через це такі дані як теми запитуються на сервері кожний раз при спробі продивитись профіль поточного користувача;

— екран редагування профілю — дозволяє редагувати дані поточного облікового запису. Фото профілю є необов’язковим, тому кожний користувач може додати або змінити фото профілю на даному екрані. Важливо відмітити, що зазвичай користувачі у якості фотографії профілю використовують зображення, зроблені на даному мобільному телефоні, і сучасні телефони оснащені камерами із великою роздільною здатністю. Наприклад, камера 13Мп створює зображення, що займатимуть близько 40Мб оперативної пам’яті у момент завантаження них на сервер, що може спровокувати виникнення StackOverflowError. Більш того, сервер даного сервісу повинен бути розрахованим на велику кількість користувачів, тому нерентабельно зберігати на сервері зображення, що займатимуть по 3-6Мб, тим більше кінцеві користувачі ніколи не будуть потребувати перегляду фото у максимальній оригінальній якості. Для вирішення цієї проблеми було написано спеціальний метод, що створює стиснену копію оригінального зображення, яке і відправляється на сервер. Зазвичай така копія займає кількасот кілобайтів на постійному сховищі сервера. Після відправлення дана копія видаляється з постійного сховища мобільного телефону. Іншою перевагою такого підходу є й те, що інтернет-з’єднання на мобільному пристрої зазвичай нестабільне та повільне, тому бажано зменшити навантаження на нього (чим менше даних для обміну, тим стабільнішою є робота інтернет-з’єднання, а отже і робота програми);

— екран коментування наділений достатньо складною логікою, тому відділений від екрану профілю. Список коментарів не приходить в одному запиті із інформацією про користувача, тому що може займати великий об’єм пам’яті, незважаючи на невисоку цінність цих даних для кінцевого користувача. Даних екран дозволяє переглядати та додавати коментарі до користувача. Коментар можна додати тільки користувачу, із яким було проведено поїздку.

До бізнес-логіки екрану перегляду профілю відноситься посередник “GetUserInfoUseCase”, що містить логіку взаємодії із методом запиту інформації про користувача за його ідентифікаційним номером репозиторія “Repository”. До бізнес-логіки списків тем відносяться посередники “GetGraduateWorkUseCase” та

“CreateGraduateWorkUseCase”, що взаємодіють із методами перегляду та додавання нових тем до профілю користувача системи.

4.6 Розробка модуля роботи з дипломами

Модуль роботи з дипломами зберігає в собі цільову логіку усього додатку, специфічну для даного сервісу.

Даний модуль є доволі складним та об’ємним, тому його було зручно поділити на чотири об’ємні модулі:

— модуль створення дипломів;

— модуль пошуку дипломів;

— модуль перегляду дипломів.

Модуль пошуку дипломів є специфічним для студентів, а два інші модулі є загальними для обох типів користувачів. Таким чином, модулі з функціоналом, специфічним для користувачів, використовують функціонал загальних обслуговуючих модулів.

4.6.1 Модуль створення дипломів

Даний модуль забезпечує функціонал, що відповідає за створення дипломів і містить один екран, що дозволяє користувачу запвнити список полів та відправити його на сервер. Дані з форми валідуються за допомогою методів класу “FieldUtils”, аналогічно до полів на екрані авторизації та інших. До полів відносяться:

— назва теми;

— опис теми;

— наукова лабораторія;

— напрям лабораторії;

— рік навчання.

Особливістю форми є те, що вона по-різному веде себе для різних типів користувачів. Так, студент може створювати теми у будь-якій лабораторії із відповідними до неї напрямами, а викладач, у свою чергу, прив’язаний до

конкретної лабораторії і не має можливості редагувати автоматично обрану лабораторію. Дана логіка реалізована у посереднику GetLecturerLabUseCase.

Бізнес-логіка даного модуля також включає в себе посередників “CreateGraduateWorkUseCase” та “GetPrioritiesByLaboratoryUseCase”, що відповідно відповідають за процес створення диплому та відображення напрямків згідно з обраною лабораторією, при цьому дані посередники працюють із методами абстрактного репозиторію “Repository” і ніяк не пов’язані із конкретними реалізаціями даних задач. Таким чином, чітко видно незалежність бізнес-логіки від зовнішнього рівня архітектури, тобто принципи стрункої архітектури дотримані.

4.6.2 Модуль пошуку дипломів

Даний модуль забезпечує функціонал, що відповідає за пошук дипломів і містить загалом три екрани (шляхи відкриття профілю викладача),та перетинається із логікою модуля профілю:

— екран списку викладачів кафедри — є важливим для користувача-студента та дозволяє йому проводити пошук цікавих для нього викладачів, після чого він має можливість переглянути їх профіль та дипломні роботи, після чого виконати певні дії. Даний екран використовує посередника “GetLecturersUseCase”;

— екрани перегляду лабораторій — дозволяють переглянути наявні на кафедрі лабораторії, а також їх напрями та склад. Загалом всі екрани зі списками є нащадками класу “ListFragment”, що відповідає за відображення абстрактного списку елементів та обробку подій натискання на них. Різниця між екранами полягає у використанні специфічного “ListUseCase” посередника та відображення за допомогою абстрактного адаптера “ListAdapter” із прив’язкою до окремого типізованого об’єкту “ViewHolder”[56]. Масив об'єктів поступає на екран, яких делегує обов’язки перетворення сутностей в елементи екрану спеціальному адаптеру “ListAdapter”, що містить методи для прив’язки даних у сутностях до елементів екрану. При чому адаптер контролює створення елементів екрану та різні події взаємодії користувача зі списком, а функції прив’язки даних сутності до елементу екрану адаптер делегує об’єкту класу “ViewHolder”, таким чином

дотримується перший принцип SOLID — принцип єдиної відповідальності, тобто кожному класу виділена окрема єдина задача, і для зміни відображення елементу маршруту розробнику не доведеться змінювати код в “ListAdapter”, а тільки в “ViewHolder”. При натисканні на елементі списку елементів відбувається перехід на наступний екран за наступною логікою:

— список лабораторій > склад лабораторії (список напрямів, список викладачів);

— список напрямів > інформація про напрям;

— список викладачів > профіль викладача.

Бізнес-логіка даного модуля включає в себе посередників “GetLabsUseCase”, та “GetLabPrioritiesUseCase”, що використовують методи репозиторія “Repository”.

4.6.3 Модуль перегляду дипломних тем

Даний модуль забезпечує функціонал, що відповідає за перегляд інформації про дипломну роботу та займає один екран. Після того як користувач відкрив профіль, він може побачити список дипломних робіт, що створив сам а також список тем, на які він підписаний. Логіка списків базується на попередньо описаному базовому класі “ListFragment”. Різниця лиш в тому що для завантаження тем використовується посередник “GetGraduateWorksUseCase”, а в якості адаптера — “GraduateWorkAdapter”. Після натискання на тему додаток перенаправляється на екран перегляду теми. На ньому можна побачити все те, що заповнюється під час створення теми. Також на даному екрані можна побачити автора роботи і список підписаних на роботу людей. Завдяки запиту до посередника “GetSubscribersUseCase”, користувач отримує інформацію про користувачів, що підписані на тему, а також статус їхньої підписки. Серед статусів є:

— на розгляді;

— відхилено;

— прийнято.

Якщо тема належить поточному користувачу-студенту, він має можливість її запропонувати. Запропонування виконується за допомогою двох посередників:

“GetLecturersByPriorityUseCase” та “OfferGraduateWorkUseCase”, де перший витягує список можливих викладачів, а другий — підписує користувача на тему.

Якщо тема не належить поточному користувачу-студенту і він бажає підписатися на тему, то на екрані перегляду дипломної роботи є можливість підписатися на тему за допомогою посередника “SubscribeGraduateWorkUseCase”.

Якщо користувач вже підписаний на тему, він має можливість відписатися від теми за допомогою посередника “UnsubscribeGraduateWorkUseCase”.

4.7 Розробка модуля графіків

Модуль графіків містить один екран, який зображає графік виконання дипломних робіт в залежності від року навчання. Дані графіку беруться з серверу за допомогою посередника “GetScheduleUseCase”, а на стороні клієнта будується html-таблиця за допомогою класу “ScheduleCreator”.

Висновки до розділу 4

Даний проект складається з кількох модулів, взаємодія кожного з яких базується на обробці дій користувача за допомогою класів-посередників, що імплементують базовий клас “UseCase”. Так як загалом кожний модуль взаємодіє із сервером, поєднуючою ланкою є модуль взаємодії з сервером.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблена система працює на операційній системі Android, тому можна просто та швидко інсталювати додаток на смартфон, який відповідає мінімальним потребам.

5.1 Інсталяція та системні вимоги

Система може працювати на смартфоні лише при наявності встановленої операційної системи Android версії не нижче 5.0 із наявними сервісами Google Play та вільним місцем на внутрішньому сховищі у розмірі 50Мб.

Для того, щоб встановити додаток на смартфон потрібно завантажити APK-файл та виконати його, але для того щоб процес був успішним, треба дозволити встановлення з невідомих джерел у налаштуваннях пристрою.

Даний додаток потребує наступні дозволи від користувача телефону:

— дозвіл на читання та запис із зовнішнього сховища;

— доступ до інтернету;

— доступ до камери.

5.2 Сценарії роботи користувача з системою

Інтерфейс програми складається з декількох робочих просторів. Кожен з них створений для керування відповідною частиною програми.

На рисунку 5.1 зображено екран авторизації. Це перший екран, який побачить користувач при запуску додатку. Вгорі екрану видно логотип кафедри, а основними кольорами додатку стали фіолетовий, жовтий та білий.

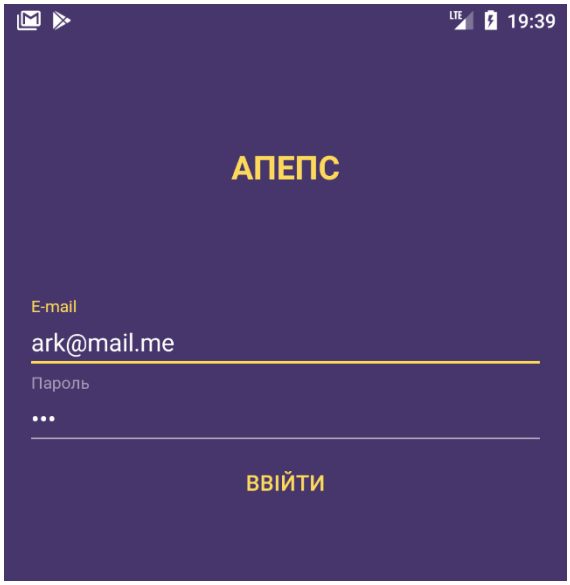

Рисунок 5.1 — Екран авторизації

Логіном слугує email користувача. Необхідним в email є символ “@”

Пароль не має обмежень за своїм складом. Єдине правило — довжина не менше шести символів. Таке обмеження є стандартним для багатьох додатків.

Якщо користувач введе некоректні значення у поле, воно підсвітиться із відповідною помилкою.

Після успішного логіну користувач потрапляє до свого профілю.

На рисунку 5.2 зображено екран перегляду профілю користувачів.

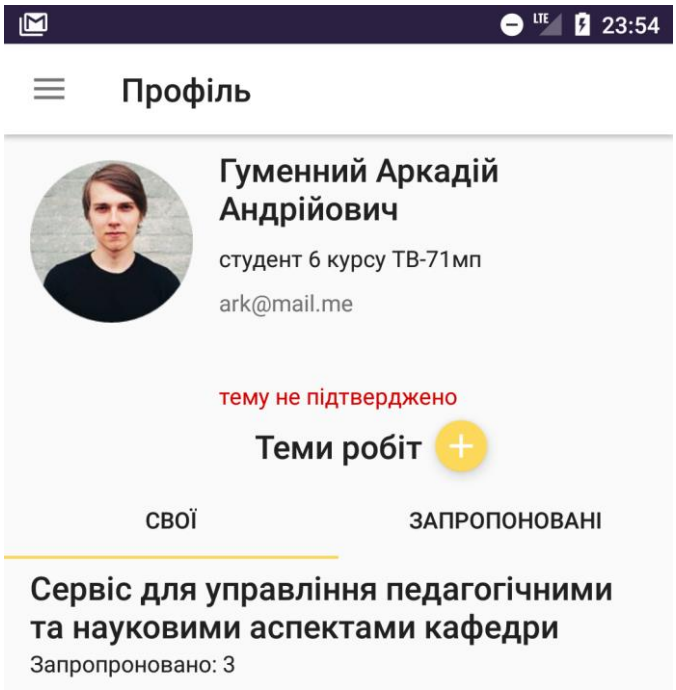

Рисунок 5.2 — Профіль користувача

На даному екрані зображена вся інформація про користувача, а саме:

— фотографія;

— ім’я та прізвище;

— статус (науковий ступінь для викладачів, група для студентів);

— список дипломних робіт (своїх та запропонованих).

Для студентів виводиться статус підтвердженності роботи. Для викладачів виводиться кількість вільних місць для дипломної роботи як показано на рисунку 5.3:

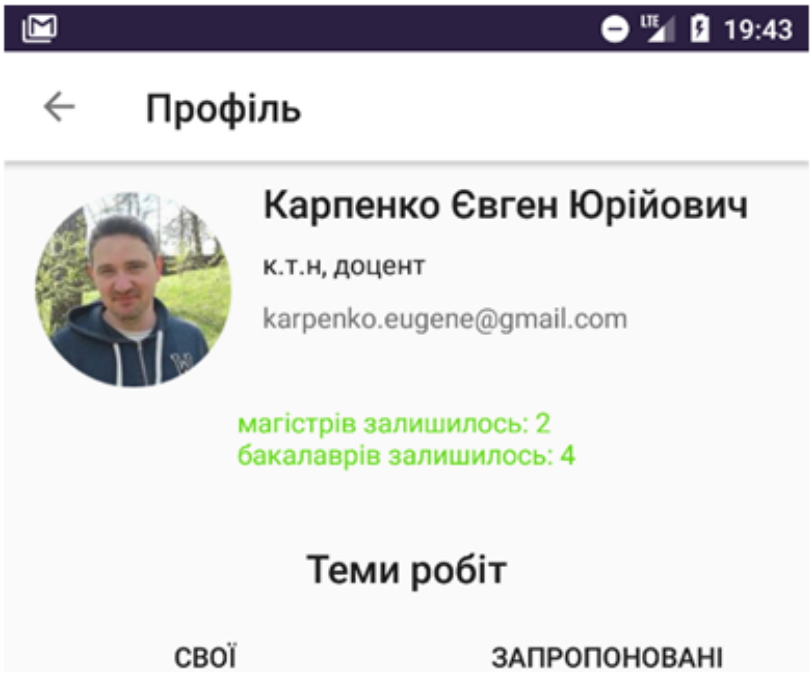

Рисунок 5.3 — Профіль користувача-викладача

Якщо профіль не належить поточному користувачу, на ньому відсутня кнопка додавання нової теми.

Повідомлення можуть бути наступними для студента:

— тему не підтверджено (червоний);

— тему підтверджено (зелений).

А для викладача наступними:

— залишилось місць: N (зелений);

— місця не залишились (червоний).

Для викладача повідомлення пишуться окремо для кожного рівня навантаження

При натисненні на кнопку “+” користувач переходить на екран створення нової теми. Екран створення нової теми зображено на рисунку 5.4.

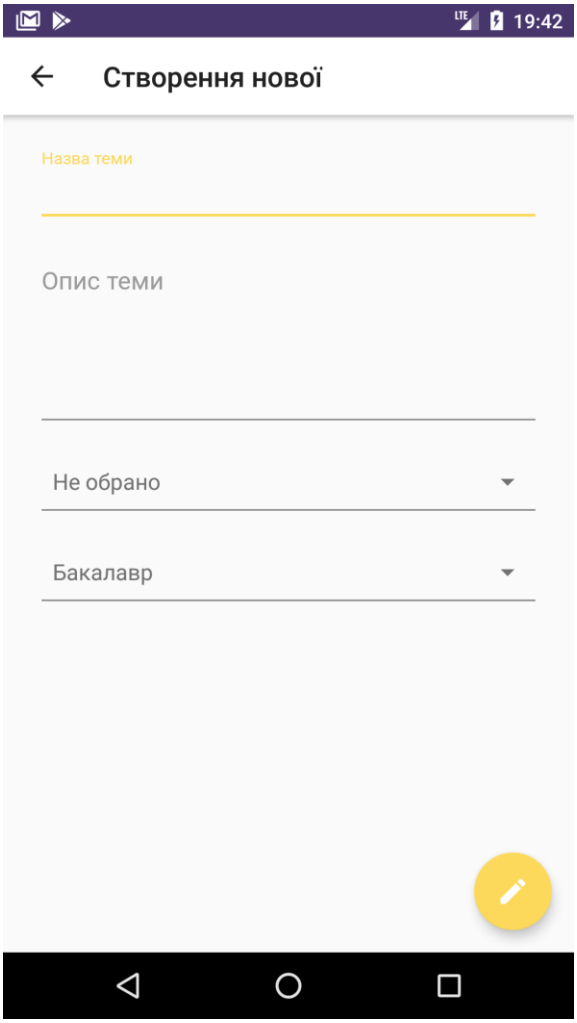

Рисунок 5.4 — Екран створення нової теми

Для створення нової теми необхідно ввести таки дані:

— тема;

— опис теми.

Також необхідно обрати лабораторію та напрям у ній, а також визначити ступінь даної роботи (бакалавр, магістр)

Важливо зазначити, що теми можуть створювати і студенти, і викладачі, але різниця в тому що викладачі можуть створювати теми тільки в рамках своїх лабораторій, а студенти – у будь-якій лабораторії.

Для визначення напрямку для роботи студент може скористатись боковим меню, у якому є дві вкладки: лабораторії та викладачі.

З екрану профілю також можна перейти на екран детального перегляду дипломної роботи (рисунок 5.5).

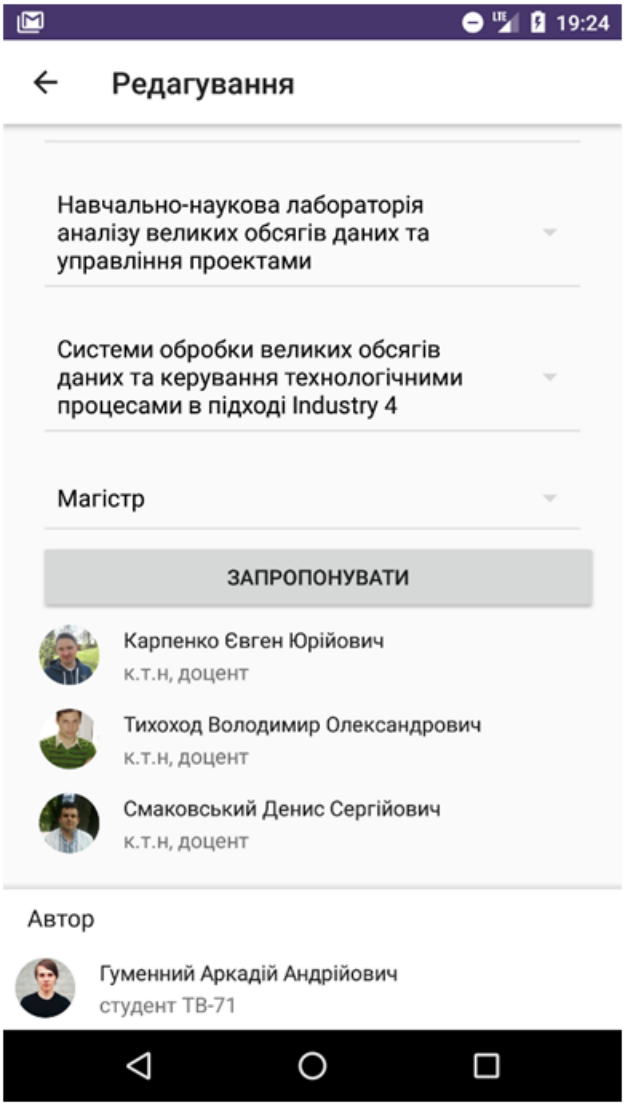

Рисунок 5.5 — Детальний перегляд теми

На даному екрані видно наступні дані:

— назва теми;

— опис теми;

— напрям та лабораторія;

— автор;

— підписники теми.

Вкладка “викладачі” (рисунок 5.6) містить список всіх викладачів за алфавітним порядком, тому студент може швидко знайти викладача, до якого хоче потрапити на наукову роботу, щоб почати взаємодіяти.

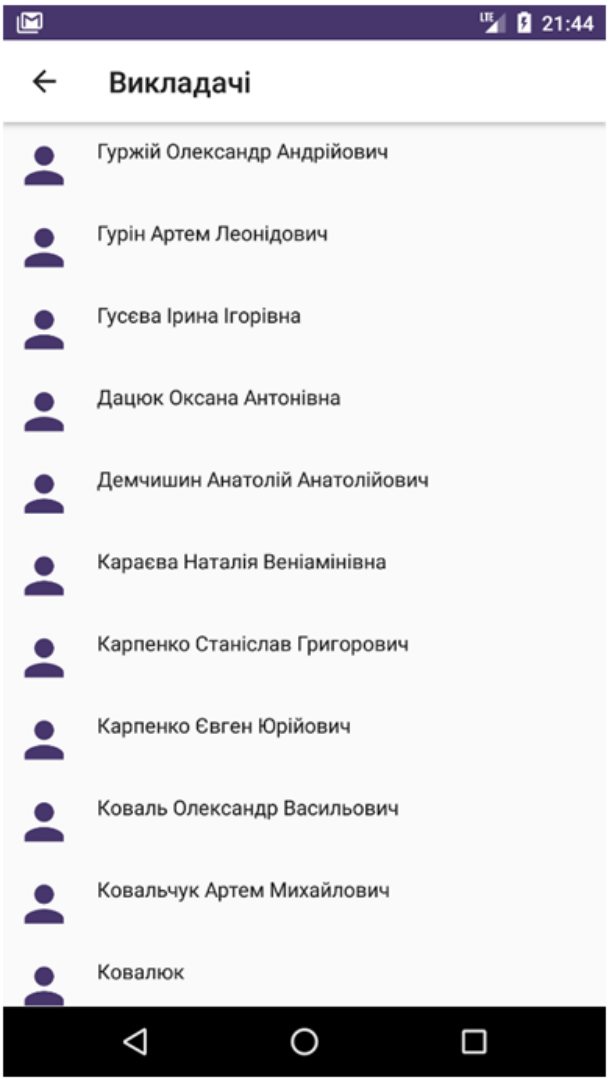

Рисунок 5.6 — вкладка “Викладачі”

З даного екрану можна перейти на профіль викладача, де можна побачити його теми та завантаженість, а також підписатись на його теми. Після підписки на тему викладач може підтвердити вас у якості виконавця. При перегляді дипломної роботи також видно усіх підписників на неї а також стан їх підписки (розглядаються, затверджені чи відхилені).

Сам викладач не має можливості запропонувати свою тему конкретному студенту. Викладач тільки створює їх і чекає на підписки, після чого розглядає їх.

Студент, у свою чергу, має можливість самостійно підписувати викладача на свою тему, тобто запропоновувати її.

Для ознайомлення з кафедрою та її лабораторіями можна перейти на вкладку “лабораторії” (рисунок 5.7).

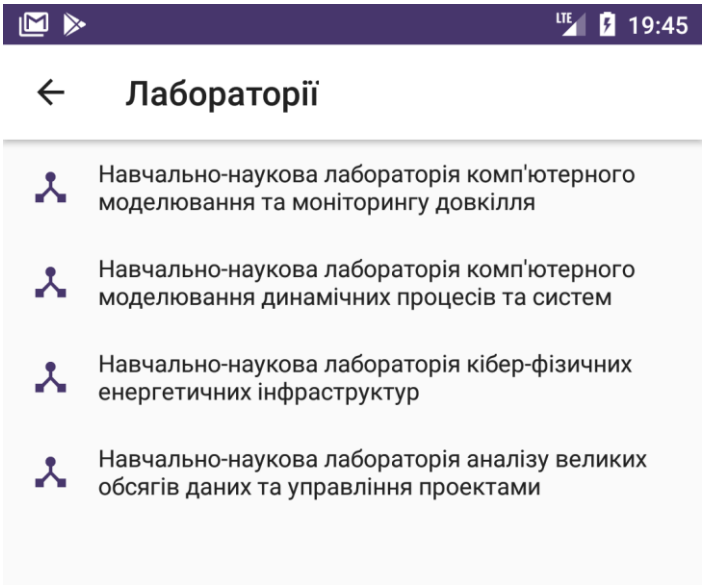

Рисунок 5.7 — вкладка “Лабораторії”

А з неї на вкладку напрямків та викладачів, що належать до даної лабораторії (рисунок 5.8)

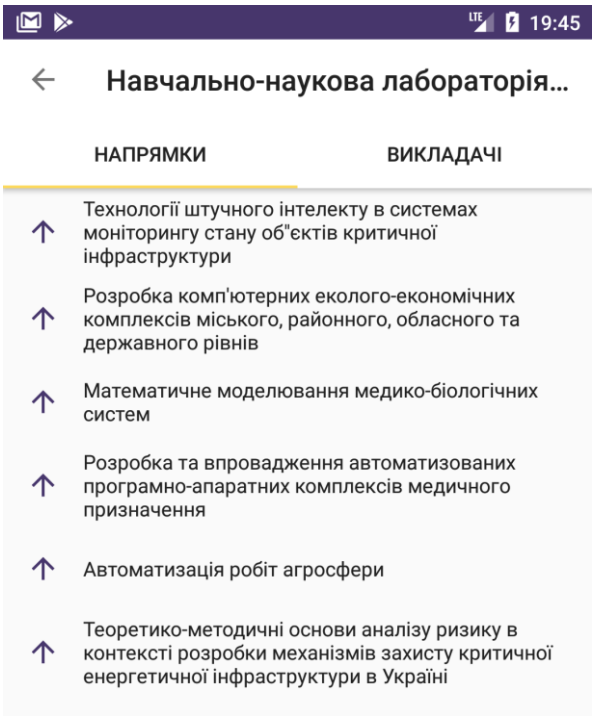

Рисунок 5.8 — вкладка “Склад лабораторії”

Користувач має можливість детальніше почитати про напрямок лабораторії на екрані “напрямок”, де є короткий опис напрямку. Вкладка “лабораторії” допомагає користувачеві визначитись із тим, куди краще віднести його тему (бо без напрямку створити тему неможливо), або навпаки підштовхнути на нові ідеї.

Також в програмі присутній екран перегляду графіків виконання робіт, що зображено на рисунку 5.9.

Рисунок 5.9 — вкладка “Графіки виконання”

Як видно з рисунку, таблиця графіку містить наступні колонки:

— перелік робіт;

— термін виконання;

— звітні документи.

Висновки до розділу 5

Дизайн додатку виконано за правилами Material Design[57], що полягає у способі організації навігації додатку (бокове меню, стрілка повернення назад, назва екрану в тулбарі), а також використанні прийнятих за стандарт розмірів та відношень елементів (списки, іконки, акцентні кольори та кнопки дії). Завдяки подібній організації додатку, користувачам android-смартфонів легко взаємодіяти із інтерфейсом даного додатку.

6 СТАРТАП ПРОЕКТ

Розділ містить проведення маркетингового аналізу стартап-проекту із ціллю визначення можливості його впровадження на ринку та способів реалізації цього впровадження. Проведення аналізу складається з кроків, що описано нижче.

6.1 Опис ідеї проекту

Опис ідеї представляється у вигляді наступних таблиць:

— зміст запропонованої ідеї;

— потенційні напрямки застосування;

— основні вигоди для користувача продукту;

— особливості продукту у порівнянні із конкурентами.

Перші три пункти подаються у вигляді таблиці 6.1 і дають цілісне уявлення про зміст ідеї та потенційні напрямки застосування.

Таблиця 6.1. Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Організація наукової та освітньої діяльності кафедри	1. Створення електронних черг	1. Уніфікація певних робочих процесів
	2. Організація дистанційного спілкування між студентом та викладачем	2. Надання актуальної інформації
	3. Документування та ведення статистики на кафедрі	3. Документальне підтвердження виконаних завдань

Аналіз потенційних техніко-економічних переваг ідеї (тобто особливості продукту) порівняно із конкурентними продуктами передбачає:

— визначення переліку техніко-економічних властивостей та характеристик ідеї;

— визначення проектів-конкурентів, що вже існують на ринку;

— збір інформації щодо значень техніко-економічних показників для ідеї даного проекту та конкурентних проектів згідно з визначеним вище переліком;

— проведення порівняльного аналізу показників із наступними значеннями: а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 6.2). [58]

Таблиця 6.2. Визначення сильних, слабких та нейтральних характеристик

No п/п		**товари/концепції конкурентів**			
		Мій проект	**Campus KPI**	**Online Tables**	**Messenger**
1	W слабка сторона	Відсутність веб-версії	Відсутність мобільної версії	Відсутність вузької направленості	Відсутність вузької направленості
2		Неможливість прикріпляти файли	Жорстка прив’язка до конкретного вузу	Немає ролей у акаунтів	Багато зайвого функціоналу
3	N нейтральна сторона	Можливість вирішення питань з дипломними роботами	Можливість бродкастити зміни на своїй сторінці	Можливість своєчасно оновлювати дані	Можливість пошуку ввикладачів що зареєстровані
4	S сильна сторона	Уніфікований простий механізм взаємодії	Велика база користувачів	Просунутий механізм для роботи з таблицями	Нотіфікації
5		Орієнтація на дві мобільні платформи	Розширений функціонал із направленням на університетську діяльність	Можливість переведення даних в локальний файл	Можливість розширеного спілкування

6.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проведено технологічний аудит, за допомогою якого можна реалізувати ідею проекту. Для визначення технологічної здійсненності ідеї проекту необхідно проаналізувати наступні складові (таблиця 6.3):

— технологія виготовлення продукту згідно з ідеєю проекту;

— пошук визначених технологій;

— визначення ступеня доступності технологій авторам проекту.

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

Таблиця 6.3. Технологічна здійсненність ідеї проекту

№ п/п	**Ідея проекту**	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
1	Нативний інтерфейс користувача	Android Studio, XCode	Наявна	Доступна безкоштовно
2	Серверна частина	Spring+Hibernate	Наявна	Доступна безкоштовно
3	Реляційна база даних	MySQL	Наявна	Доступна безкоштовно
4	Реалізація чистої архітектури	Dagger, RxJava, Kotlin, Retrofit, RxSwift, Swift	Наявна	Доступна за вільною ліцензією Apache, MIT
5	Сервіси для нотіфікацій	Firebase	Наявна	Доступна за гроші згідно з тарифними планами

Висновок: проект реалізувати можливо.

Обрана технологія реалізації ідеї проекту: Нативний інтерфейс користувача, реалізація серверної частини та чистої архітектури, без нотіфікацій.

6.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, необхідних та наявних під час впровадження проекту на ринку, а також ринкових загроз, які можуть цьому перешкодити, необхідне для того щоб спланувати напрями розвитку проекту із урахуванням стану ринку (включаючи потреби потенційних клієнтів та пропозиції конкурентних проектів).

Для цього в першу чергу необхідно провести аналіз попиту: наявність, обсяг, та динаміка розвитку ринку (таблиця 6.4).

Таблиця 6.4. Попередня характеристика потенційного ринку стартап-проекту

No п/п	**Показники стану ринку (найменування)**	**Характеристика**
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	2000 грн
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Правила Вузів
6	Середня норма рентабельності в галузі (або по ринку), %	60 %

Середня норма рентабельності в галузі (або по ринку) порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проект.

За результатами аналізу таблиці робиться висновок щодо того, чи є ринок привабливим для входження за попереднім оцінюванням.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 6.5).

Таблиця 6.5. Характеристика потенційних клієнтів стартап-проекту

No п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Налагодження взаємодії між викладачами та студентами	ВУЗИ та інші учбові заклади	особливості купівлі: компанії заключають довготривалі договори, а стартапери віддають перевагу пробному терміну стандарти: в компаніях архітектором баз даних та мобільними розробниками може виступати декілька людей, в стартапах зазвичай це одна людина	стабільність роботи Невисока ціна (корпоративна закупівля ліцензії на цілий відділ) Наявність документації Підтримка мобільних платформ

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 6.6-5.7).

Таблиця 6.6. Фактори загроз

No п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1	Підходить для кафедр з орієнтацією на лабораторії	Для кафедр з іншою структурою необхідні зміни та доповнення	Додавання альтернативного групування (не за лабораторіями)
2	Власний формат зберігання	При необхідності змінити інструмент, компанії доведеться це робити вручну, оскільки	Додавання можливості автоматизованого експорту в різні типи сховищ

Таблиця 6.6 (Продовження)

		використовується власна структура збереження даних	
3	Обмеженість функцій	Інструмент обмежений наявними функціями і не має деяких функцій, які мають конкуренти (наприклад: індекси)	Додавання нових функцій за потреби

Таблиця 6.7. Фактори можливостей

No п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1	Популярність мобільних платформ	Мобільна індустрія наразі друга за розвитком після веб-індустрії в постійно набирає обертів	Вихід на мобільний ринок
2	Потреба у автоматизації процесів	Ідея з’явилась через існуючі потреби на кафедрі. Вона актуальна скрізь	Надання інструменту для організації наукової роботи кафедри
3	Відсутність повноцінних альтернатив	Існуючі альтернативи не надають можливості контролювати процес ведення дипломів безпосередньо	Реалізація відсутнього функціоналу

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку. Аналіз пропозиції необхідно виконати аналізуючи існуючі види конкуренцій.

Пропозиції повинні відповідати на питання “Як просувати продукт”.

Аналіз пропозицій зображено на таблиці 6.8.

Таблиця 6.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста	чиста	Прямі договори з стартапами, презентація продукту на виставках
2. За рівнем конкурентної боротьби - локальний/національний /...	національний	Публікація статей на міжнародних сайтах
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	внутрішньогалузева	Розвивати напрямки, нерозвинуті конкурентами
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	товарно-видова	Розповідати про свої переваги перед конкурентом у цій галузі
5. За характером конкурентних переваг - цінова / нецінова	нецінова	Надання функцій, які не надають конкуренти
6. За інтенсивністю - марочна/не марочна	марочна	Надання функцій, які не надають конкуренти

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 6.9). [59]

Таблиця 6.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники (Архітектори БД)**	**Клієнти (Розробники)**	**Товари-замінники**

Таблиця 6.9 (Продовження)

	Campus	Excel online, Messenger	Мінімізація витрат часу постачальників	Контроль якості	Лояльність споживачів
Висновки:	Визначити інтенсивність конкурентної боротьби з боку прямих конкурентів	Є можливості виходу на ринок, оскільки існуючі рішення не надають потрібних переваг	Постачальники підлаштовуються під ринок	Клієнти диктують вимоги згідно з умовами експлуатації	Обмеження для роботи на ринку через товари замінники

На основі аналізу конкуренції, проведеного в п. 3.5 (таблиця 6.9), а також із урахуванням характеристик ідеї проекту (таблиця 6.2), вимог споживачів до товару (таблиця 6.5) та факторів маркетингового середовища (таблиця 6.6-6.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею 6.10. [60]

Таблиця 6.10. Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1	Підтримка декількох платформ	Існуючі конкуренти орієнтовані на веб
2	Орієнтація на вузьку галузь	Існуючі конкуренти не орієнтовані на вирішення питань, що є пріоритетними для даного проекту

За визначеними факторами конкурентоспроможності (таблиця 6.10) проводиться аналіз сильних та слабких сторін стартап-проекту (таблиця 6.11). [61]

Таблиця 6.11. Порівняльний аналіз сильних та слабких сторін

No п/п	**Фактор конкурентоспроможності**	**Бали 1-20**	**Рейтинг товарів-конкурентів у порівнянні з Database Generator (даним продуктом)**						
			-3	**-2**	**-1**	**0**	**1**	**2**	**3**
1	Підтримка декількох платформ	20	+						
2	Генерація додаткових зручних методів	10			+				

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних та слабких сторін, загроз та можливостей (таблиця 6.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 6.11). [62]

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища.

Таблиця 6.12. SWOT-аналіз стартап-проекту

Сильні сторони: Орієнтація на декілька мобільних платформ	**Слабкі сторони:** Відсутність нотифікацій
Можливості: Популярність мобільних платформ Потреба в урегулюванні робочих процесів на кафедрі Відсутність повноцінних альтернатив	**Загрози:** Низький рівень кастомізації Обмеженість функцій

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. [63]

Таблиця 6.13. Альтернативи ринкового впровадження стартап-проекту

No п/п	**Альтернатива (орієнтовний комплекс заходів) ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1	Орієнтація поточної моделі на ринок стартаперів	10 %	40 год
2	Орієнтація поточної моделі на ринок державних установ	70 %	240 год
3	Орієнтація поточної моделі на ринок ентерпрайз	60 %	160 год
4	Переорієнтація на генерацію серверної частини	80 %	120 год
5	Переорієнтація на веб-розробку	60 %	100 год
6	Переорієнтація на перенесення БД з одних платформ на інші	60 %	60 год

Альтернатива, де отримання ресурсів є більш простим та ймовірним – №2 " Орієнтація поточної моделі на ринок державних установ ", що становить 70 відсотків. Це значення перевищує інші альтернативи.

Альтернатива, де строки реалізації є більш стислими – №1 " Орієнтація поточної моделі на ринок стартаперів". Терміни реалізації в цьому разі становлять лише 40 годин.

6.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 6.14).

За результатами аналізу потенційних груп споживачів автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар. Також на основі цих даних визначається стратегія охоплення ринку. [64]

Таблиця 6.14. Вибір цільових груп потенційних споживачів

No п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота входу у сегмент**
1	Стартапери	Готові	Високий	Середня	Дуже складно
2	Державні установи	Потребують переговорів	Середній	Низька	Дуже складно
3	Ентерпрайз	Потребують переговорів	Середній	Низька	Дуже складно
Які цільові групи обрано: державні установи					

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 6.15).

Таблиця 6.15. Визначення базової стратегії розвитку

No п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкурентос-проможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***
1	Орієнтація поточної моделі на ринок державних установ	Стратегія концентрованого маркетингу	Державні установи діють за застарілими схемами і тільки в деяких областях з’являється автоматизація процесів	Стратегія диференції

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту.

Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 6.16).

Таблиця 6.16. Визначення базової стратегії конкурентної поведінки

No п/п	**Чи є проект «першопрохідцем» на ринку?**	**Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?**	**Чи буде компанія копіювати основні характеристики товару конкурента, і які?**	**Стратегія конкурентної поведінки***
1	Так	шукати нових споживачів	Ні	Стратегія заняття конкурентної ніші

З обраних сегментів до постачальника (державні установи) та до продукту розробляється стратегія позиціонування (таблиця 6.17). що полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати торгівельну марку/проект. [65]

Таблиця 6.17. Визначення стратегії позиціонування

No п/п	**Вимоги до товару цільової аудиторії**	**Базова стратегія розвитку**	**Ключові конкуренто-спроможні позиції власного стартап-проекту**	**Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)**

Таблиця 6.17 (Продовження)

1	Стабільність роботи Невисока ціна (корпоративна закупівля ліцензії на цілий відділ) Наявність документації Підтримка мобільних платформ	Стратегія диференціації	Державні установи діють за застарілими схемами і тільки в деяких областях з'являється автоматизація процесів	Пришвидшення робочих процесів підтримка декількох платформ

6.5 Розроблення маркетингової програми стартап-проекту

Для цього у таблиці 6.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару. [66]

Таблиця 6.18. Визначення ключових переваг концепції потенційного товару

No п/п	**Потреба**	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)**
1	Пришвидшення робочих процесів	Вузьконаправлена спеціалізація	Більшість конкуренти не мають шаблону для організації процесів
2	Підтримка мобільних платформ	Підтримка Android, iOS, наявний гнучкий Rest API	Прямий конкурент орієнтований на веб

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 6.19).

Таблиця 6.19. Опис трьох рівнів моделі товару

Рівні товару	**Сутність та складові**

Таблиця 6.19 (Продовження)

<table>
<tr><td>I. Товар за задумом</td><td colspan="3">Додаток для організації наукових та освітніх аспектів кафедри</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>можливість оптимізації витрат часу</td><td>М</td><td>Тл</td></tr>
<tr><td>можливість оптимізації витрат коштів</td><td>М</td><td>Вр</td></tr>
<tr><td>відповідність актуальним технологіям</td><td>Нм</td><td>Тх</td></tr>
<tr><td colspan="3">Відповідає вимогам ДСТУ ISO/IEC 25030:2015 Програмна інженерія. Вимоги щодо якості та оцінювання програмного продукту</td></tr>
<tr><td colspan="3">Пакування: готовий до використання apk-файл</td></tr>
<tr><td colspan="3">Марка: К412</td></tr>
<tr><td>III. Товар із підкрі-пленням</td><td colspan="3">Потенційний користувач може ознайомитись з поточним товаром з наукових конференцій та публічних виступів, а також наукових вісників на яких була представлена інформація про даний продукт</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: Назва і контент захищені ліцензією; захист інтелектуальної власності</td></tr>
</table>

М/Нм – монотонні або немонотонні;

Вр/Тх/Тл/Е/Ор – вартісні, технічні, технологічні, ергономічні або органолептичні (останній – для продуктів харчування)

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару. [67]

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (таблиця 6.20).

Таблиця 6.20. Визначення меж встановлення ціни

No п/п	**Рівень цін на товари-замінники**	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни на товар/послугу**
1	125…300 грн	240…400 грн	12000…42000 грн	125…240 грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 6.21):

— проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

— вибір та обґрунтування оптимальної глибини каналу збуту;

— вибір та обґрунтування виду посередників.

Таблиця 6.21. Формування системи збуту

No п/п	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
1	Клієнт повинен надаватися в режимах “тріал” та “повний”. Сплата має проходити через Google Play	Легість в встановленні, легкість в сплаті послуг	4: Розробник даного продукту - Веб-сайт – Google Play - Користувач.	Проводити збут силами посередника Google Play

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 6.22).

Таблиця 6.22. Концепція маркетингових комунікацій

No п/п	**Специфіка поведінки цільових клієнтів**	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
1	Купляють програми через авторизовану мережу Google Play	Google Play, веб-сайти	підтримка декількох платформ Організація робочих процесів	Довести, що програмний продукт полегшить роботу кафедри	Все що треба - тут

Висновки до розділу 6

Розроблений програмний продукт має переваги над існуючими конкурентами та є конкурентноздатним на ринку. Програма має шляхи подальшого розвитку, визначені маркетингові стратегії та шляхи збуту. Основна цільова аудиторія – це державні установи, для яких важливо поступово налагодити організацію багатьох процесів, що на даний момент виконуються за застарілими схемами.

ВИСНОВКИ

Розроблено онлайн-сервіс для організації наукової роботи кафедри. Для реалізації цього було: проведено аналіз предметної області та виділено сутності системи; проаналізовано вже існуюче програмне забезпечення та модулі що відповідають функціям системи, обрано засоби розробки; реалізовано функції системи.

В процесі ведення диплому було досліджено сучасні засоби, методи, патерни та принципи розробки програмного забезпечення, а також вдосконалено їх шляхом поєднання в шаблонну архітектуру, яка є універсальною для будь-якого android-додатку, бо спрямована на нівелювання специфічних для android-фреймворку архітектурних недоліків.

Було створено програмний продукт, що не має аналогів в Україні. Завдяки впровадженню даного програмного продукту можна буде назавжди відмовитися від застарілих неефективних засобів з організації науково-педагогічної діяльності на кафедрі. Програма має шляхи подальшого розвитку, визначені маркетингові стратегії та шляхи збуту. Основна цільова аудиторія – це студенти та викладачі вузів.

Важливо також зазначити, що обидва модулі (клієнт та сервер) написано згідно з сучасними стандартами у сфері програмного забезпечення, опираються на принципи та технології, що використовуються у багатьох відомих проектах.

Як результат, продукт є розширюваним, а отже представлений функціонал може бути збагачений новими функціями у майбутньому без перешкод, а також бути легким у підтримці, що є запорукою успішності будь-якого проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коржов В. Многоуровневые системы клиент-сервер / Валерий Коржов. – Москва: Издательство Открытые системы, 1997. – 200 с.
2. Campus KPI [Електронний ресурс] — Режим доступу: http://login.kpi.ua/.
3. Гриффитс Д. Headfirst Android / Дон Гриффитс. – San-Francisco: Headfirst, 2014. – 540 с.
4. Feipeng L. Android Native Development Kit Cookbook / Liu Feipeng., 2015. – 312 с.
5. Hartson R. The UX Book: Process and Guidelines for Ensuring a Quality User Experience / R. Hartson, P. Pyla., 2008. – 175 с.
6. Bennet J. Xamarin in Action: Creating native cross-platform mobile apps / Jim Bennet., 2017. – 654 с.
7. Eisenman B. Learning React Native: Building Native Mobile Apps with JavaScrip / Bonnie Eisenman., 2018. – 670 с.
8. Goransson A. Efficient Android Threading: Asynchronous Processing Techniques for Android Applications / Anders Goransson., 2016. – 350 с.
9. Schwarz R. The Android Developer's Cookbook: Building Applications with the Android SDK / Ronan Schwarz.. – 546 с.
10. Smyth N. Android Studio 3.0 Development Essentials - Android 8 Edition / Neil Smyth., 2018. – 254 с.
11. Wolfson M. Android Developer Tools Essentials: Android Studio to Zipalign / M. Wolfson, D. Felker., 2016. – 438 с.
12. Ratabouil S. Android NDK Beginners Guide / Sylvain Ratabouil., 2012. – 343 с.
13. Krochmalski J. IntelliJ IDEA Essentials / Jaroslaw Krochmalski. – San-Francisco: Packt publishing, 2014. – 358 с.
14. Kotlin [Електронний ресурс] — Режим доступу: https://kotlinlang.org/.
15. Эккель Б. Философия Java. Полное издание: Java / Брюс Эккель. — СПб: Питер, 2003. — 831 с.

16. Gradle [Електронний ресурс] — Режим доступу: https://gradle.org/.
17. Odersky M. Programming in Scala / M. Odersky, L. Spoon. – New York: Artima, 2015. – 965 с.
18. Square [Електронний ресурс] – Режим доступу до ресурсу: http://square.github.io/.
19. Jemerov D. Kotlin in Action / D. Jemerov, S. Isakova. – New York: Manning, 2016. – 563 с.
20. THE WORST MISTAKE OF COMPUTER SCIENCE [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.lucidchart.com/techblog/2015/08/31/the-worst-mistake-of-computer-science/.
21. NullPointerException [Електронний ресурс]. – 2000. – Режим доступу до ресурсу: https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html.
22. Jones S. Implementing Functional Prog Languages / Simon Jones., 2001. – 421 с.
23. Kotlin Properties [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://kotlinlang.org/docs/reference/properties.html.
24. Kotlin Backing field [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://kotlinlang.org/docs/reference/backing-field.html
25. Jin B. Designing Web APIs: Building APIs That Developers Love / B. Jin, S. Sahni. – New York: O'reilly, 2015. – 336 с.
26. Kotlin Lazy delegate [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://kotlinlang.org/docs/reference/property-delegate.html
27. Kotlin delegate [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://kotlinlang.org/docs/reference/delegate.html
28. Kotlin extension functions [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://kotlinlang.org/docs/reference/functions.html
29. Albahari J. C# 7.0 in a Nutshell: The Definitive Reference / J. Albahari, B. Albahari. – New York: O'reilly, 2017. – 563 с.
30. Sobkowicz M. Learn Game Programming with Ruby: Bring Your Ideas to Life with Gosu / Mark Sobkowicz. – London: Programmatic programming, 2014. – 455 с.

31. Gamma E. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm. – Boston: Addison-Wesley, 2001. – 657 с.
32. Spring [Електронний ресурс] — Режим доступу: https://spring.io/.
33. Coward D. Java EE 7: The Big Picture / Danny Coward. – Chicago: Oracle Press, 2010. – 457 с.
34. Inversion of control [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring.
35. Jacobson I. Aspect-Oriented Software Development with Use Cases / Ivar Jacobson. – Boston: Ad, 2004. – 302 с.
36. JDBC [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/technetwork/java/javase/jdbc/index.html.
37. Working with NoSql [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-nosql.html.
38. MVC [Електронний ресурс] – Режим доступу до ресурсу: https://yiiframework.com.ua/ru/doc/guide/basics.mvc/.
39. Hunter J. Java Servlet Programming: Help for Server Side Java Developers / J. Hunter, W. Crawford. – New York: O'reilly, 2007. – 547 с.
40. Hibernate ORM [Електронний ресурс] — Режим доступу: http://hibernate.org/orm/.
41. Ray E. Learning XML / Erik Ray. – New York: O'reilly, 2004. – 322 с.
42. RxJava [Електронний ресурс] — Режим доступу: https://github.com/ReactiveX/RxJava.
43. Reactive Manifesto [Електронний ресурс] – Режим доступу до ресурсу: https://www.reactivemanifesto.org/.
44. Retrofit [Електронний ресурс] — Режим доступу: http://square.github.io/retrofit/.
45. What is HTTP request? [Електронний ресурс] – Режим доступу до ресурсу: http://toolsqa.com/client-server/http-request/.
46. Miles R. Learning UML 2.0: A Pragmatic Introduction to UML / R. Miles, K. Hamilton. – New York: O'reilly, 2009. – 295 с.

47. Болье А. Learning SQL [Електронний ресурс] / Алан Болье. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.
48. Exciting secrets about MVVM that nobody tells you [Електронний ресурс] – Режим доступу до ресурсу: https://android.jlelse.eu/exciting-secrets-about-mvvm-that-nobody-tells-you-a95548ea684b.
49. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert Martin. – New York: Prentice Hall, 2000. – 468 с.
50. Freeman E. Head First Design Patterns: A Brain-Friendly Guide / E. Freeman, B. Bates. – San-Francisco: Headfirst, 2002. – 586 с.
51. Hall G. M. Adaptive Code: Agile coding with design patterns and SOLID principles / Gary McLean Hall. – Washington: Microsoft Press, 2002. – 314 с.
52. Osis J. Topological UML Modeling: An Improved Approach for Domain Modeling and Software Development / J. Osis, U. Donins. – London: Elsevier, 2004. – 267 с.
53. Newman S. Building Microservices: Designing Fine-Grained Systems / Sam Newman. – New York: O'reilly, 2009. – 688 с.
54. Repository pattern [Електронний ресурс] – Режим доступу до ресурсу: https://deviq.com/repository-pattern/.
55. JSON syntax [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/js/js_json_syntax.asp.
56. RecyclerView.ViewHolder [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ViewHolder.
57. Material Design [Електронний ресурс] – Режим доступу до ресурсу: https://material.io/design/.
58. SWOT-аналіз [Електронний ресурс] // Навчальні матеріали онлайн – Режим доступу до ресурсу: http://pidruchniki.com/1577111551903/marketing/swot-analiz.
59. Модель п'яти сил конкуренції за М. Портером [Електронний ресурс] – Режим доступу до ресурсу: http://stud.com.ua/45490/ekonomika/.

60. Цибульов П. М. Управління інтелектуальною власністю : монографія/ Цибульов П. М., Чеботарьов В. П., Зінов В. Г. , Суїні Ю., за ред. П. М. Цибульова. – К. : «К. І. С.», 2005. – 448 с.
61. Квашнин А. Как управлять портфелем технологий и интеллектуальной собственностью : серия методических материалов «Практические руководства для центров коммерциализации технологий» / под рук. П. Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 60 с.
62. Квашнин А. Как продвигать проекты коммерциализации технологий : серия методических материалов «Практические руководства для центров коммерциализации технологий» / М. Катешова, А. Квашнин, под рук. П. Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 52 с.
63. Петруненков А. Оценка коммерческой привлекательности проекта [Электронный ресурс] // Технологический бизнес. – 1999. – № 2. Режим доступа:http://www.techbusiness.ru/tb/archiv/number2/page01.htm
64. Тиль, П. От нуля к единице : как создать стартап, который изменит будущее / П. Тиль, Б. Мастерс; перевод с англ. – Москва : Альпина паблишер, 2015. – 188 с.
65. Харниш, В. Правила прибыльных стартапов : как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва : Манн, Иванов и Фербер, 2012. – 279 с.
66. Экланд С. Ангелы, драконы и стервятники : как привлечь правильных инвесторов в свой стартап и сохранить бизнес / С. Экланд ; пер. с англ. О. Терентьевой. – Москва : Манн, Иванов и Фербер, 2011. – 275 с.
67. Маллинс, Дж. Поиск бизнес-модели : как спасти стартап, вовремя сменив план / Дж. Маллинс, Р. Комисар ; пер. с англ. М. Пуксант и Е. Бакушевой. – Москва : Манн, Иванов и Фербер, 2012. – 329 с.

ДОДАТОК А

Публікації

Android додаток управління педагогічними та науковими аспектами кафедри

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3252_18М

Аркушів 17

2018

ДОДАТОК Б

Акт впровадження

Android додаток управління педагогічними та науковими аспектами кафедри

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3252_18М

Аркушів 2

2018